DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD		BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB	UUU UUU UUU	GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
--	--	--	---	--

DV

DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD	BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB	GGGGGGGG GG GG GG GG GG GG GG GG GG GG	PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	RRRRRRRR RRRRRRRR RR RR RR RR RR RR RRRRRR	\$	
		\$					

89012345678901234567890123456789012345678901234567

BEGIN

0189

0190

COPYRIGHT (c) 1978, 1980, 1982, 1984 BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. ALL RIGHTS RESERVED.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.

WRITTEN BY Bert Beander February, 1982

MODIFIED BY Rich Title

Ping Sager

Brad Becker

Walter Carrell III

Added code and tables needed to support BLISS, C, MACRO.
Added code and tables needed to support PASCAL.
Update DBG\$GL_CURRENT_PRIMARY for self_referential records
Added tables and support for Built-in functions.

MODULE FUNCTION
This module contains the language-independent lexical scanner and parser used to scan and parse both address expressions and language expressions for all languages. It also contains all the parse tables needed to scan and parse each of the languages supported by DEBUG.

REQUIRE 'SRC\$: DBGPROLOG.REQ';

LIBRARY 'LIB\$:DBGGEN.L32';

FORWARD ROUTINE

AAA_DUMMY: NOVALUE,

DBG\$ADDR_EXP_INT,

DBG\$BUILD_PRIMARY_SUBNODE: NOVALUE,

DBG\$EXP_INT,

! Dummy routine--does nothing, not used ! Address Expression Interpreter driver ! Build a Primary Descriptor Sub-Node ! Expression Interpreter driver routine

D

DB

DBGPARSER V04-000			C 14 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1
204 205 206 207 208 209 210 211 212 213 214 215 216 217 218	0336 1 0337 1 0338 1 0339 1 0340 1 0341 1 0342 1 0344 1 0345 1 0346 1 0347 1 0347 1 0348 1 0349 1	SAVED_TOKEN: INITIAL(0), STATE_TABLE: REF NUMST\$TABLE, SUBSCRIPT_TERM_TBL:	Token pointer saved by Primary Parser until next call by Expr. Parser Pointer to the Number Scanner State Table for the current language Pointer to Terminator Table to use when picking up subscripts Pointer to Predefined Identifier Table Terminator code for last terminator token found by lexical scanner Length in characters of last terminator found by the lexical scanner Stack of Variant Set RST Entry pointers Stack of RST Variant Entry pointers Stack of variant component indicies Current index for VARSTACKN stacks

Page 5 (2)

DE

DE V

MACROS TO GENERATE PARSE TABLES

These macros are used to generate the parse tables used by the Lexical Scanner and the Parser to parse both language-independent and language-specific constructs accepted by DEBUG.

CHARACTER TABLE

Define the macros which generate the Character table. This includes the macros which generate the base Character Table for language UNKNOWN and the macros which generate the Character Exception Tables for the individual languages. The Character Exception Table for a language lists those characters which have different characteristics in that language that the default characteristics specified for language UNKNOWN. The Character Table for a language is thus formed by copying the default table for language UNKNOWN and then overlaying the entries for those characters listed in the language's Character Exception Table.

These macros are used as follows. To generate the default Character Table, the following sequence of macro invocations is used:

CHAR_TABLE(TBLNAME, CHAR_ENT(CHAR, CLASS, BIT1, BIT2, ...) CHAR_ENT(CHAR, CLASS, BIT1, BIT2, ...);

Here TBLNAME is the name of the table to be built. The CHAR_TABLE macro defines the whole Character Table as a BLOCKVECTOR and causes each element in the vector to be filled with zeroes unless this is overridden by an explicit CHAR_ENT invocation for that specific character.

The CHAR_ENT macro sets the character characteristics for a specified character. Here CHAR is the character itself (e.g., 'A'), CLASS is the Number Scanner Character Class, and BIT1, BIT2, ... are the names of the other characteristics bits to be set for this character. The CLASS parameter is automatically prefixed by "NUMST\$K_CLASS_" by the macro. Also, the BITn parameters are automatically prefixed by "CHRTBL\$M_" to generate the proper mask value names.

To generate a Character Exception Table for a language, the following macro invocations are used:

CHAR_EXCEPTION TABLE (TBLNAME, CHAR_ENTRY(CHAR, CLASS, BIT1, BIT2, ...)

CHAR_ENTRY(CHAR, CLASS, BIT1, BIT2, ...);

Here CHAR, CLASS, and BIT1, BIT2, ... have the same meanings as for the CHAR_ENT macro above. Similarly, TBLNAME is the name given to the new Character Exception Table. The only difference between these macros and those described above is that a different data structure is generated.

DE

```
MACRO
                CHAR_TABLE (TBLNAME) =
                                      OWN TBLNAME: VECTOR[256,LONG] PSECT(DBG$PLIT) PRESET(%REMAINING) %;
                           MACRO
                                CHAR_ENT(CHAR, CLASS) = [CHAR]=((%NAME('NUMST$K_CLASS_',CLASS)^4) OR CHAR_FLAGS(%REMAINING)) %;
                           MACRO
                                CHAR_FLAGS[FLAG] = ",FLAG) %;
                                CHAR EXCEPTION TABLE (TBLNAME) =
                                      %THEN
                                           BIND TBLNAME = PLIT(REP 0 OF (0)): VECTOR[,LONG]
                                     BIND TBLNAME = PLIT(%REMAINING): VECTOR[,LONG]
                           MACRO
                                CHAR_ENTRY(CHAR, CLASS) = CHAR^24 OR %NAME('NUMST$K_CLASS_',CLASS)^4 OR CHAR_FLAGS(%REMAINING) %;
                             Define fields of Character Exception Table entry. These definitions are only used in the DBG$PARSER_SET_LANGUAGE routine below.
                          ! Character characteristics bits ! The character itself
                           MACRO
                                CE_ENTRY = BLOCK[1, WORD] FIELD(CE_FLDS) %;
                             OPERATOR TABLE
                             Define the macros which generate an Operator Table for a language. An Operator Table is a counted vector (a PLIT) of longwords which point to Operator Lexical Token Entries for the operators of the language. Each pointer is relative to the address TABLEBASE so that the code is completely Position-
                              Independent (PIC); the true pointer value is thus the longword value plus
                              TABLEBASE.
                              An Operator Table is generated with the following macro invocations:
                                  OPERATOR_TABLE (TBLNAME,
                                     OPERATOR_ENTRY (OPNAME, CODE, KIND, LEFT_PREC, RIGHT_PREC, FLAG1,...),
                                     OPERATOR_ENTRY(OPNAME, CODE, KIND, LEFT_PREC, RIGHT_PREC, FLAG1,...));
```

thus the longword value plus TABLEBASE.

DI

D

Page

(3)

....

Page

....

.....

TERMINATOR ENTRY (NAMESTRING, CODE) = UPLIT BYTE(0,

0692

(3)

NUMBER SCANNER STATE TABLE

Define the macros which generate Number Scanner State Tables. These tables are used in the lexical scanning of numeric constants and are in general specific to each language.

A Number Scanner State Table for a language is generated with the following macro invocations:

NUMBER_STATE_TABLE (TBLNAME,

NUMBER_STATE (STATE ID, NUMBER_TRANSITION (CHAR_CLASS, ACTION, NEXTSTATE),

NUMBER_TRANSITION(CHAR_CLASS, ACTION, NEXTSTATE), NUMBER_TRANSITION(OTHER, ACTION, NEXTSTATE)),

NUMBER_STATE (STATE ID, NUMBER_TRANSITION (CHAR_CLASS, ACTION, NEXTSTATE),

NUMBER_TRANSITION(CHAR_CLASS, ACTION, NEXTSTATE), NUMBER_TRANSITION(OTHER, ACTION, NEXTSTATE));

The NUMBER_STATE_TABLE macro sets up the Number Scanner State Table as a whole and binds the table name TBLNAME to that structure. Each state in the table is declared with the NUMBER_STATE macro whose STATE ID argument names the state. The actual state name is prefixed by 'NUMST\$XX_STATE_' by the macro and must be declared as such in the COMPILETIME declaration below.

The NUMBER_TRANSITION macro defines one transition from the current state to some other state. The transition is taken if the next input character is of the character class specified by CHAR_CLASS. (CHAR_CLASS is automatically prefixed by 'NUMST\$K_CLASS' to generate the class code constant.) If the transition is taken, the action specified by ACTION is taken. ACTION, which is automatically prefixed by 'NUMST\$K_ACT_' by the macro, is a CASE index used by the Number Scanner to select an action routine which performs whatever semantic processing is appropriate. After the action routine executes, the next state in the state table is given by NEXTSTATE. NEXTSTATE is automatically prefixed by 'NUMST\$K_STATE_ by the macro and must be declared as the STATE_ID on some other NUMBER_STATE declaration in the state table.

Every state in the state table must have at least two transitions and the

! Current index into primary parser state

COMPILETIME

PRIMARY\$XX_CUR_LOC = 0,

```
DBGPARSER
V04-000
                                                                                                                     16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                             0864
0865
0866
0867
0868
0870
0871
0872
0873
0876
0877
0878
0879
0880
                                                                                                                        Index values in primary parser state
     PRIMARY$XX_STATE_START_STATE = 0,
PRIMARY$XX_STATE_GET_GEOBAL = 0,
PRIMARY$XX_STATE_GOT_BACKSLASH = 0,
PRIMARY$XX_STATE_GOT_DOT = 0,
PRIMARY$XX_STATE_GOT_SUBSCRIPT = 0,
PRIMARY$XX_STATE_GOT_SUBSCRIPT2 = 0,
PRIMARY$XX_STATE_GOT_BRACKET = 0,
PRIMARY$XX_STATE_GOT_DEREF = 0,
PRIMARY$XX_STATE_GOT_DOT_SLASH = 0,
PRIMARY$XX_STATE_END_STATE = 0;
table vector for the states
                                                The following data structure is used in the SAVE_SUBSCRIPTS routine
                                               and the PATHNAME_TO_PRIMARY routine.
                                        1 FIELD
                                                   SUBSCR_DESC_FIELDS =
                                                          SUBSCR$B_PATH_INDEX = [0,0,8,0],
                                                                                                                                       Says which pathname element
                             0884
0885
0886
0887
0888
0889
                                                                                                                                            the subscripts were
                                                                                                                                             associated with
                                                          SUBSCR$B_SUBCNT = [0,8,8,0],

SUBSCR$V_RANGE = [0,16,1,0],

SUBSCR$V_ASTER = [0,17,1,0],

SUBSCR$V_MARKER = [0,18,1,0],

SUBSCR$L_LBOUND = [1,0,32,0],

SUBSCR$L_UBOUND = [2,0,32,0],
                                                                                                                                       Count of subscripts so far
                                                                                                                                       True if range was specified 
True for "*" range
                                                                                                                                       Marker bit (see SAVE_SUBSCRIPTS)
                             0890
0891
0893
0893
0894
0895
0896
0896
0896
0901
0903
0904
0906
0907
0916
0917
0918
0919
0919
0919
                                                                                                                                       Lower bound of range
                                                                                                                                       Upper bound of range
                                                          TES:
                                            MACRO
                                                   SUBSCR$DESC = BLOCKVECTOR[DBG$K_PATHNAME_SIZE.3,LONG]
                                                                                           FIELD(SUBSCR_DESC_FIELDS)%;
                                           LITERAL
                                                   SUBSCR_DESC_SIZE = 3*DBG$K_PATHNAME_SIZE*%UPVAL;
                                               PARSER FLAGS
                                               Define parser flags which may have different settings for different languages. These flags are stored in the Table of Language-Specific Tables and are copied
                                                to individual OWN variables when the current language is set. These flags
                                                control the behavior of the parser in those cases where different languages
                                               have different behaviors and the easiest way of encoding these differences is to have a flags to control the parser. The following flags are available:
                                                          MULTIPLE_SUBSCR - The language allows multiple subscript parentheses in array references. For example, PASCAL allows X[1,2,3,4] to be written as X[1,2][3][4]. This flag should not be set if the second form is not
                                                                                           equivalent to the first form of array reference.
                                               Each of these flags is set to TRUE or FALSE in an OWN variable with the same
                                              name as the flag name give here.
```

DB

DBGPARSER V04-000	C 15 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 18
## 898	UPLIT BYTE (PERCENT IDENT, %ASCIC 'XR',) - TABLEBASE, UPLIT BYTE	

DE V

DI

```
E 15
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                                        VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                        1142
1143
P 1144
1145
1146
P 1147
1148
; 1012
; 1013
; 1014
; 1015
; 1016
; 1017
                                                       TERMINATOR_ENTRY(',', TERM_COMMA));
                                         TERMINATOR_TABLE(EQUAL_TERM_TBL,
TERMINATOR_ENTRY('=', TERM_EQUAL, BALANCED_PARENS));
                                         TERMINATOR_TABLE(DO_TERM_TBL, TERM_DO, BALANCED_PARENS));
                        P 1150
   1019
   10223456789012334567100667810068
                                         TERMINATOR_TABLE(THEN_TERM_TBL; TERM_THEN, BALANCED_PARENS));
                       1151
1152
P 1153
P 1154
1155
                                         TERMINATOR_TABLE(COMCOL_TERM_TBL,
TERMINATOR_ENTRY(',', TERM_COMMA),
TERMINATOR_ENTRY(':', TERM_COLON, MUST_BE_SINGLE));
                           1156
                                         P
                        P 1158
P 1159
                            1160
                            1161
                       P 1162
1163
1164
P 1165
                                         TERMINATOR_TABLE(OPEN_TERM_TBL,
TERMINATOR_ENTRY('T', TERM_OPEN));
                                         TERMINATOR_TABLE(COMPAREN_TERM_TBL,
TERMINATOR_ENTRY(',', TERM_COMMA),
TERMINATOR_ENTRY(')', TERM_CLOSE, BALANCED_PARENS));
                        P 1166
1167
                            1168
                        P 1169
                                         TERMINATOR_TABLE(TO_TERM_TBL, TERM_TO, BALANCED_PARENS));
                           1170
                       P 1172
P 1173
1174
1175
                                         TERMINATOR_TABLE(BY_TERM_TBL,
TERMINATOR_ENTRY('BY', TERM_BY, BALANCED_PARENS),
TERMINATOR_ENTRY('DO', TERM_DO, BALANCED_PARENS));
                                         TERMINATOR_TABLE(BIT_SELECT_TERM_TBL,
TERMINATOR_ENTRY(',', TERM_COMMA),
TERMINATOR_ENTRY('>', TERM_GTRTHAN));
                        P 1176
P 1177
                           1178
1179
                          1180
1181
1182
1183
1184
1185
                                         TERMINATOR_TABLE(SET_CONSTANT_TERM_TBL,
TERMINATOR_ENTRY(',', TERM_COMMA),
TERMINATOR_ENTRY(')', TERM_CLOSE, BALANCED_PARENS),
TERMINATOR_ENTRY('...', TERM_DOT));
                        P
                            1186
1187
                                             Generate a table indexed by Terminator Code which has pointers to the above Terminator Lexical Token Entry tables. This table is used in DBG$EXP_INT
                                             and DBG$ADDR_EXP_INT to look up the terminator table to be used for the
                            1189
1190
1191
1192
1193
1194
1195
1196
1197
                                             current expression of address expression.
                                                TERM_POINTER_TBL: VECTOR[TOKENSK_MAX_TERMINATOR + 1]
PSECT(DBGSPLIT) PRESET(
                                                       [TOKEN$K_TERM_NONE]
[TOKEN$K_TERM_COMMA]
[TOKEN$K_TERM_EQUAL]
[TOKEN$K_TERM_DO]
[TOKEN$K_TERM_THEN]
                                                                                              = EMPTY TERM TBL
= COMMA TERM TBL
= EQUAL TERM TBL
= DO TERM TBC
                                                                                                                                          - TABLEBASE,
                                                                                                                                          - TABLEBASE,
                                                                                                                                          - TABLEBASE,
                                                                                                                                          - TABLEBASE,
                            1198
                                                                                               = THEN_TERM_TBL
                                                                                                                                          - TABLEBASE,
```

D

Page 20 (4)

```
DBGPARSER
V04-000
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   (4)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           Page
                                                                                                                                                                                                                                                          [TOKEN$K_TERM_COMCOL] = COMCOL_TERM_TBL

[TOKEN$K_TERM_CMWHDO] = CMWHDO_TERM_TBL

[TOKEN$K_TERM_OPEN] = OPEN_TERM_TBL

[TOKEN$K_TERM_COMPAREN] = COMPAREN_TERM_TBL

[TOKEN$K_TERM_TO] = TO_TERM_TBL

[TOKEN$K_TERM_BY] = BY_TERM_TBL
              1069
1070
1071
1072
1073
1074
                                                                                                         - TABLEBASE,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  TABLEBASE,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 TABLEBASE,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 TABLEBASE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 TABLEBASE);
              1075
1076
1077
                                                                                                                                                                                                         Generate the base Character Table. The Character Table for each supported language is generated by using this Character Table as a base and then specifying a list of exceptions for specific characters. The result is a language-specific Character Table, generated in the CHARTBL vector.
                1078
                1079
              1080
1081
1082
1083
1084
1085
1086
1087
1088
1090
1091
1093
1094
1095
                                                                                                                                                                                  | a language-specific Character Table, generated in the CHARTBL vector.
| CHAR_ENT('A', HEXDIGIT, ALPHABETIC, IDENT_ANYWHERE), CHAR_ENT('B', B, ALPHABETIC, IDENT_ANYWHERE), TERMINATOR), CHAR_ENT('C', HEXDIGIT, ALPHABETIC, IDENT_ANYWHERE), CHAR_ENT('C', HEXDIGIT, ALPHABETIC, IDENT_ANYWHERE), CHAR_ENT('F', HEXDIGIT, ALPHABETIC, IDENT_ANYWHERE), CHAR_ENT('F', HEXDIGIT, ALPHABETIC, IDENT_ANYWHERE), CHAR_ENT('F', HEXDIGIT, ALPHABETIC, IDENT_ANYWHERE), CHAR_ENT('I', OTHER, ALPHABETIC, IDENT_ANYWHERE), CHAR_ENT('I', OTHER, ALPHABETIC, IDENT_ANYWHERE), CHAR_ENT('I', OTHER, ALPHABETIC, IDENT_ANYWHERE), CHAR_ENT('I', OTHER, ALPHABETIC, IDENT_ANYWHERE), CHAR_ENT('N', OTHER, ALPHABETIC, IDENT_ANYWHERE), CHAR_ENT('N', OTHER, ALPHABETIC, IDENT_ANYWHERE), CHAR_ENT('O', OTHER, ALPHABETIC, IDENT_ANYWHE
            1096
1097
1098
1099
           1100
           1102
           1104
            1106
1107
1108
1109
                                                                                                                                                                                                                                                     CHAR ENT('a',
CHAR ENT('b',
CHAR ENT('c',
CHAR ENT('d',
CHAR ENT('e',
CHAR ENT('f',
CHAR ENT('m',
CHAR ENT('n',
CHAR ENT('p',
CHAR ENT('p',
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 IDENT_ANYWHERE).
IDENT_ANYWHERE).
IDENT_ANYWHERE.
IDENT_ANYWHERE.
IDENT_ANYWHERE).
                                                                                                                                                                                                                                                                                                                                                                      HEXDIGIT, ALPHABETIC, B, ALPHABETIC,
            1110
                                                                                                                                                                                                                                                                                                                                                                 B, ALPHABETIC, ALPHABETIC, ALPHABETIC, ALPHABETIC, ALPHABETIC, ALPHABETIC, ALPHABETIC, OTHER, ALPHABETIC, ALPHABETIC, OTHER, ALPHABETIC, ALPHABETIC, OTHER, ALPHABETIC,
            1111
            1112
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                TERMINATOR),
            1114
            1115
            1116
              1117
             1118
            1119
            1120
1121
1122
1123
1124
1125
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   IDENT_ANYWHERE),
```

```
G 15
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
       DBGPARSER
V04-000
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  VAX-11 Bliss-32 V4.0-742 [DEBUG.SRCJDBGPARSER.B32;1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    CHAR_ENT('q',
CHAR_ENT('r',
CHAR_ENT('s',
CHAR_ENT('t',
CHAR_ENT('u',
CHAR_ENT('w',
CHAR_ENT('w',
CHAR_ENT('x',
CHAR_ENT('z',
CHAR_ENT('z',
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      ALPHABETIC, IDENT_ANYWHERE),
ALPHABETIC, IDENT_ANYWHERE),
ALPHABETIC, IDENT_ANYWHERE,
ALPHABETIC, IDENT_ANYWHERE,
ALPHABETIC, IDENT_ANYWHERE),
ALPHABETIC, IDENT_ANYWHERE),
ALPHABETIC, IDENT_ANYWHERE,
ALPHABETIC, IDENT_ANYWHERE),
ALPHABETIC, IDENT_ANYWHERE),
ALPHABETIC, IDENT_ANYWHERE),
ALPHABETIC, IDENT_ANYWHERE),
ALPHABETIC, IDENT_ANYWHERE),
                                                                                                                                                                                                                                                                                                                                                                OTHER.
11289
11278
11278
11278
11278
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
11333
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    OTHER.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               OTHER,
OTHER,
OTHER,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          OTHER,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            CHAR_ENT('0', DIGIT, CHAR_ENT('1', DIGIT, CHAR_ENT('2', DIGIT, CHAR_ENT('3', DIGIT, CHAR_ENT('4', DIGIT, CHAR_ENT('5', DIGIT, CHAR_ENT('6', DIGIT, CHAR_ENT('7', DIGIT, CHAR_ENT('8', DIGIT, CHAR_ENT('8', DIGIT, CHAR_ENT('9', DIGIT, CHAR_ENT(
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      DIGIT, IDENT_MIDDLE, IDENT_END, NUMBER_START),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         DIGIT, IDENT_MIDDLE, IDENT_END, NUMBER_START),

SPACE), ! Tab character
NOTHING),
STRING QUOTE),
NOTHING),
IDENT MIDDLE, IDENT_END),
NOTHING),
STRING_QUOTE),
OPCHAR, ADDRESS_OP, TERMINATOR),
OPCHAR, ADDRESS_OP, TERMINATOR),
OPCHAR, ADDRESS_OP),
OPCHAR, ADDRESS_OP),
TERMINATOR),
OPCHAR, ADDRESS_OP),
NUMBER_START, OPCHAR, ADDRESS_OP, SPECIAL_SYMBOL),
OPCHAR, ADDRESS_OP),
TERMINATOR),
NOTHING),
ADDRESS_OP),
TERMINATOR),
TERMINATOR),
NOTHING),
OPCHAR, ADDRESS_OP),
NOTHING),
OPCHAR, SPECIAL_SYMBOL),
NOTHING),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            CHAR_ENT('9', DIGIT,

CHAR_ENT('9', OTHER,

CHAR_ENT('!', OTHER,

CHAR_ENT('!', OTHER,

CHAR_ENT('%', OTHER,

CHAR_ENT('%', OTHER,

CHAR_ENT('%', OTHER,

CHAR_ENT('%', OTHER,

CHAR_ENT('%', OTHER,

CHAR_ENT('', OTHER,

                                                     1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1177
                                                       1180
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   NOTHING));
```

999

LANGUAGE UNKNOWN PARSE TABLES

This section contains all Lexical Scanner and Parser tables for language UNKNOWN. Language "UNKNOWN" constitutes the language facilities made available by DEBUG when the actual source language is not known to DEBUG. The main use for language UNKNOWN is to provide a basic level of DEBUG support for new languages which are not yet explicitly supported by DEBUG with its own syntax and semantics.

(Language UNKNOWN comes first because some of the other languages make references to the UNKNOWN number table. The rest of the languages are in alphabetical order.)

Define the language UNKNOWN Character Table.

```
CHAR_EXCEPTION_TABLE(UNKNOWN_CHARTBL,
CHAR_ENTRY('&', OTHER, OPCHAR),
CHAR_ENTRY('/', OTHER, OPCHAR, OPCHAR_INFIX, ADDRESS_OP),
CHAR_ENTRY('<', OTHER, OPCHAR, OPCHAR_INFIX, ADDRESS_OP),
CHAR_ENTRY('=', OTHER, OPCHAR, OPCHAR_INFIX, TERMINATOR),
CHAR_ENTRY('>', OTHER, OPCHAR, OPCHAR_INFIX, TERMINATOR),
CHAR_ENTRY('A', OTHER, OPCHAR, SPECIAL_SYMBOL),
CHAR_ENTRY('L', OTHER, OPCHAR),
CHAR_ENTRY('L', OTHER, OPCHAR),
CHAR_ENTRY('L', OTHER, TERMINATOR));
```

Define the language UNKNOWN Operator Table for operators whose names are identifiers.

```
OPERATOR TABLE (UNKNOWN IDENT OPTBLE OPERATOR ENTRY ('EQL', EQU' OPERATOR ENTRY ('NEQ', NOT OPERATOR ENTRY ('GTR', GTR' OPERATOR ENTRY ('GEQ', GTR' OPERATOR ENTRY ('LSS', LSS' OPERATOR ENTRY ('NOT', NOT OPERATOR ENTRY ('AND', AND OPERATOR ENTRY ('OR', OR OPERATOR ENTRY ('XOR', XOI OPERATOR ENTRY ('YOR', XOI OPERATOR ENTRY ('EQV', EQ')
                                                                                                                                                                                                                                                                            50).
50).
50).
50).
50).
40).
30).
10);
                                                                                                                                                 EQUAL
                                                                                                                                                                                                                                                            50.
50.
50.
200.
200.
10.
                                                                                                                                                NOT_EQUAL,
GTR_THAN,
GTR_EQUAL,
LSS_THAN,
LSS_EQUAL,
                                                                                                                                                                                                                          INFIX,
                                                                                                                                                                                                                          INFIX,
                                                                                                                                                                                                                          INFIX,
                                                                                                                                                                                                                          INFIX,
                                                                                                                                                                                                                         INFIX,
PREFIX,
                                                                                                                                                 NOT,
                                                                                                                                                  AND.
                                                                                                                                                                                                                          INFIX.
                                                                                                                                                                                                                        INFIX.
INFIX.
INFIX.
                                                                                                                                                  OR,
                                                                                                                                                  XOR,
                                                                                                                                                 EQV.
```

Define the language UNKNOWN Operator Table for operators whose names are composed of operator characters such as "+", " or "*". This table includes operators which are part of Primary Symbols (such as "\" and ".").

```
OPERATOR_TABLE(UNKNOWN_OPCHAR_OPTBL,
OPERATOR_ENTRY('\', GLOBAL_SLASH, PREFIX, 0, 0, PRIMARY),
OPERATOR_ENTRY('\', BACKSLASH, INFIX, 0, 0, PRIMARY),
OPERATOR_ENTRY('\', DOT, INFIX, 0, 0, PRIMARY),
OPERATOR_ENTRY('\', SUBSCRIPT, POSTFIX,0, 0, PRIMARY),
OPERATOR_ENTRY('\', SUBSCRIPT, POSTFIX,0, 0, PRIMARY),
```

```
I 15
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
                                                                                                                                                                                                                                                                                                                             VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32;1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                 Page
V04-000
     OPERATOR_ENTRY('^',
                                                                                                                                                                                                          PASCAL_DEREF,
                                                                                                                                                                                                                                                                   POSTFIX,0, 0, PRIMARY),
                                                                                                                 OPERATOR ENTRY('%',
OPERATOR ENTRY('%',
OPERATOR ENTRY('*',
OPERATOR ENTRY('',
OPERATOR ENTRY
                                                                                                                                                                                                          CONCATENATE,
                                                                                                                                                                                                                                                                    INFIX,
                                                                                                                                                                                                                                                                                                 60).
60).
70).
70).
80).
50).
50).
                                                                                                                                                                                                                                                                     INFIX,
                                                                                                                                                                                                          SUBTRACT,
UNARY_PLUS,
UNARY_MINUS,
                                                                                                                                                                                                                                                                   INFIX,
PREFIX,
PREFIX,
INFIX,
INFIX,
                                                                                                                                                                                                          MULTIPLY,
                                                                                                                                                                                                          DIVIDE
                                                                                                                                                                                                          POWER_OF,
                                                                                                                                                                                                                                                                     INFIX,
                                                       1380
1381
1382
1383
1384
1385
1386
                                                                                                                                                                                                          EQUAL
                                                                                                                                                                                                                                                                    INFIX,
                                                                                                                                                                                                         NOT_EQUAL,
NOT_EQUAL,
GTR_THAN,
GTR_EQUAL,
LSS_THAN,
LSS_EQUAL,
OPENPAREN,
                                                                                                                                                                                                                                                                     INFIX.
                                                                                                                                                                                                                                                                     INFIX.
                                                                                                                                                                                                                                                                     INFIX.
                                                                                                                                                                                                                                                                     INFIX,
                                                                                                                                                                                                                                                                     INFIX.
                                                                                                                                                                                                                                                                   INFIX.
PREFIX.
POSTFIX.
                                                                                                                                                                                                                                                                                               200.
                                                                                                                                                                                                                                                                                                                  200. LEXICAL);
                                                         1389
1389
1390
                                                                                                                                                                                                          CLOSEPAREN.
                                                1391
1392
1393
1394
P 1395
P 1396
P 1397
P 1398
                                                                                             Define the UNKNOWN Terminator Lexical Token Table for subscript expressions.
                                                                                            Here we allow subscript expressions to be terminated by ")" (end of subscripts) and by "," (more subscripts to follow).
                                                                                    TERMINATOR TABLE (UNKNOWN SUBSCR TERM TBL,
TERMINATOR ENTRY(')', TERM_CLOSE, BALANCED_PARENS),
TERMINATOR ENTRY(')', TERM_CLOSE, BALANCED_PARENS),
TERMINATOR ENTRY(':', TERM_COLON, MUST_BE_SINGLE),
TERMINATOR ENTRY(',', TERM_COMMA));
                                                         1400
                                                         1401
                                                       1402
1403
1404
1405
1406
1407
1408
1410
1411
1412
1413
1414
1415
1416
                                                                                            Define the language UNKNOWN Number Scanner State Table. This is a finite-state
                                                                                             machine in which each transition is of the form:
                                                                                                                  NUMBER_TRANSITION(character-class, action-index, next-state)
                                                                                            where the character-class and action-index names are automatically prefixed by "NUMST$K_CLASS_" or "NUMST$K_ACT_" by the NUMBER_TRANSITION macro.
                                                                                      NUMBER_STATE_TABLE (UNKNOWN_NUMBER_TABLE,
                                                                                                   NUMBER_STATE(START_STATE,
NUMBER_TRANSITION(DIGIT, GO_PAST_DIGIT, ACCUM_INT),
NUMBER_TRANSITION(DOT, MARK_DEC_PT, LEADING_DOT),
NUMBER_TRANSITION(OFHER, NOT_NUMBER, END_STATE)),
                                                                                                   NUMBER STATE (LEADING DOT, NUMBER TRANSITION (DIGIT, GO PAST FRAC, ACCUM FRAC).
                                                        1418
                                                                                                                   NUMBER_TRANSITION(OTHER, NOT_NUMBER, END_STATE)),
                                                        1420
1421
1422
1423
1424
1425
                                                                                                   NUMBER_STATE(ACCUM_INT,
NUMBER_TRANSITION(DIGIT, GO_PAST_DIGIT, ACCUM_INT),
NUMBER_TRANSITION(DOT, MARK_DEC_PT, ACCUM_FRAT),
NUMBER_TRANSITION(HEXDIGIT, DO_NOTHING, ACCUM_HEX),
                                                 P
                                                  P
                                                                                                                   NUMBER_TRANSITION(B, DO_NOTHING, ACCUM_HEX),
```

```
VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
                                                                               NUMBER_TRANSITION(D, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(E, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(OTHER, GOT_NUMBER, END_STATE)),
NUMBER_TRANSITION(DIGIT, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(HEXDIGIT, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(B, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(D, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(E, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(E, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(OTHER, GOT_NUMBER, END_STATE)),
                                                                   NUMBER_TRANSITION(DIGIT, GO_PAST_FRAC, ACCUM_FRAC),
NUMBER_TRANSITION(DOT, BACKUP_PTRS, END_STATE),
NUMBER_TRANSITION(E, MARK_E_EXP, GET_EXPONENT),
NUMBER_TRANSITION(D, MARK_D_EXP, GET_EXPONENT),
NUMBER_TRANSITION(G, MARK_G_EXP, GET_EXPONENT),
NUMBER_TRANSITION(Q, MARK_G_EXP, GET_EXPONENT),
NUMBER_TRANSITION(Q, MARK_G_EXP, GET_EXPONENT),
NUMBER_TRANSITION(OTHER, GOT_NUMBER, END_STATE)),
                                   NUMBER_STATE(GET_EXPONENT,
NUMBER_TRANSITION(DIGIT, DO NOTHING, ACCUM_EXP),
NUMBER_TRANSITION(PLUS, DO NOTHING, GET_EXP_SIGN),
NUMBER_TRANSITION(MINUS, DO NOTHING, GET_EXP_SIGN),
NUMBER_TRANSITION(OTHER, BACKUP_PTRS, END_STATE)),
                                                                    NUMBER_STATE(GET_EXP_SIGN,
NUMBER_TRANSITION(DIGIT, DO_NOTHING, ACCUM_EXP),
NUMBER_TRANSITION(OTHER, GOT_NUMBER, END_STATE)),
                                                                    NUMBER_STATE(ACCUM_EXP,
NUMBER_TRANSITION(DIGIT, DO_NOTHING, ACCUM_EXP),
NUMBER_TRANSITION(OTHER, GOT_NUMBER, END_STATE)),
                                    1460
1461
1462
1463
                                                                    NUMBER_STATE(END_STATE, NUMBER_TRANSITION(OTHER, GIVE_ERROR, END_STATE)));
                              1464
1465
1466
1467
1468
1469
1470
1471
1473
1474
1475
1476
P 1478
P 1480
P 1481
P 1482
                                                               Define the language UNKNOWN Predefined Identifier Table.
                                                          PRID_TABLE(UNKNOWN_PRID_TABLE);
                                                               Define the language UNKNOWN Built-in function Table.
                                                          BUILT_IN_FUNCTION_TABLE(UNKNOWN_FUNCTION_TABLE);
                                                               Define the Primary Parser State Table for language UNKNOWN.
                                                          PRIMARY_STATE_TABLE (UNKNOWN_PRIMARY_TABLE,
                                                                    PRIMARY STATE (START STATE,
PRIMARY TRANSITION (GLOBAL SLASH, START GBL, GET GLOBAL),
PRIMARY TRANSITION (BACKSLASH, START SLASH, GOT BACKSLASH),
PRIMARY TRANSITION (INVOCNUM, SLASH INVOCNUM, GOT BACKSLASH),
```

```
K 15
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
 DBGPARSER
V04-000
                                                                                                                                                                                                                                                                                                                                                                                                                                                    VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
                                                                                                                                                               PRIMARY_TRANSITION(DOT, START_DOT, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, START_SUBSCR, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PASCAL_DEREF, START_DEREF, GOT_DEREF),
PRIMARY_TRANSITION(PRIMARY_TERM, START_TERM, END_STATE)),
         135567890123668901237756789012388901239990133999013399901339990
                                                                                1483
1484
1485
1486
1487
1488
1491
1493
1494
1495
PRIMARY_STATE (GET_GLOBAL
                                                                                                                                                               PRIMARY_TRANSITION(PRIMARY_TERM, GBL_TERM, END_STATE)),
                                                                                                                                         PRIMARY STATE (GOT BACKSLASH,
PRIMARY TRANSITION (BACKSLASH, SLASH SLASH, GOT BACKSLASH),
PRIMARY TRANSITION (INVOCNUM, SLASH INVOCNUM, GOT BACKSLASH),
PRIMARY TRANSITION (DOT, SLASH DOT, GOT DOT),
PRIMARY TRANSITION (SUBSCRIPT, SLASH SUBSCR, GOT SUBSCRIPT),
PRIMARY TRANSITION (PASCAL DEREF, SLASH DEREF, GOT DEREF),
PRIMARY TRANSITION (PRIMARY TERM, SLASH TERM, END STATE)),
                                                                                1496
                                                                                 1498
                                                                                                                                          PRIMARY_STATE(GOT_DOT,
PRIMARY_TRANSITION(DOT, DOT_DOT, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, DOT_SUBSCR, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PASCAL_DEREF, DOT_DEREF, GOT_DEREF),
PRIMARY_TRANSITION(PRIMARY_TERM, DOT_TERM, END_STATE)),
                                                                                 1499
                                                                                1500
                                                                    P 1501
P 1502
P 1503
P 1506
P 1506
P 1507
P 1510
P 1511
P 1513
P 1516
P 1512
P 1523
P 
                                                                                                                                          PRIMARY_STATE(GOT_SUBSCRIPT,
PRIMARY_TRANSITION(DOT, SUBSCR_DOT, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, SUBSCR_SUBSCR, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PASCAL_DEREF, SUBSCR_DEREF, GOT_DEREF),
PRIMARY_TRANSITION(PRIMARY_TERM, SUBSCR_TERM, END_STATE)),
                                                                                                                                         PRIMARY_STATE(GOT_DEREF,
PRIMARY_TRANSITION(DOT, DEREF_DOT, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, DEREF_SUBSCR, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PASCAL_DEREF, DEREF_DEREF, GOT_DEREF),
PRIMARY_TRANSITION(PRIMARY_TERM, DEREF_TERM, END_STATE)),
                                                                                                                                           PRIMARY_STATE(END_STATE));
                                                                                                                               finally define the table of pointers to the parse tables for language UNKNOWN.
                                                                                                                       LANGUAGE_TABLES(LANGUAGE = UNKNOWN,
                                                                                                                                                                                                        CHARTBL = UNKNOWN CHARTBL
                                                                                                                                                                                                      IDENT OPTBL = UNKNOWN IDENT OPTBL, OPCHAR OPTBL = UNKNOWN OPCHAR OPTBL,
                                                                                                                                                                                                      NUMBER_TABLE = UNKNOWN_NUMBER_TABLE
                                                                                                                                                                                                      PRIMARY TABLE = UNKNOWN PRIMARY TABLE
                                                                                                                                                                                                      SUBSCR_TERMS = UNKNOWN_SUBSCR_TERM_TBL,
PRIDTBE = UNKNOWN_PRID_TABLE,
                                                                                                                                                                                                       BIF_TABLE = UNKNOWN_FUNCTION_TABLE);
```

CONCATENATE,

EQUAL.

INFIX.

```
DBGPARSER
V04-000
                                                                                                                                       16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                                         VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                                                                                                                                                                                                                                                     Page
   1460
1461
1462
1463
1464
1465
1466
1467
                            P 1588
P 1589
P 1590
P 1591
1592
1593
                                                                  OPERATOR_ENTRY('/=',
OPERATOR_ENTRY('<',
OPERATOR_ENTRY('<=',
OPERATOR_ENTRY('>',
OPERATOR_ENTRY('>=',
                                                                                                                     NOT EQUAL,
LSS THAN,
LSS EQUAL,
GTR THAN,
GTR EQUAL,
                                                                                                                                                        INFIX,
                                                                                                                                                                                    20).
                                                                                                                                                                          20,
                                                                                                                                                        INFIX.
                                                                                                                                                        INFIX.
                                                                                                                                                        INFIX.
                                                     Define an Operator Lexical Token Entry for the ADA Tick operator "This token is actually used to represent a collection of postfix tick operators ("'FIRST", "LAST", ...). The lexical scanner will fill in the subcode field which identifies which tick operator was
   1469
1470
1471
1472
1473
                                 1598
1599
1600
                                                      given.
                                  1601
   1474
                                 1602
                                                          ADA_TICK_TOKEN = OPERATOR_ENTRY("", ADA_TICK, POSTFIX, 0, 0, PRIMARY);
   1476
                                 1604
                                                         ADA_TICK_TABLE: VECTOR [TOKEN$K_TICK_MAX+1] PSECT(DBG$PLIT) PRESET

[TOKEN$K_TICK_CONSTRAINED] = UPLIT (%ASCIC 'CONSTRAINED')

[TOKEN$K_TICK_FIRST] = UPLIT (%ASCIC 'FIRST')

[TOKEN$K_TICK_LAST] = UPLIT (%ASCIC 'LAST')

[TOKEN$K_TICK_LENGTH] = UPLIT (%ASCIC 'LENGTH')

[TOKEN$K_TICK_POS] = UPLIT (%ASCIC 'POS')

[TOKEN$K_TICK_PRED] = UPLIT (%ASCIC 'SIZE')

[TOKEN$K_TICK_SIZE] = UPLIT (%ASCIC 'SIZE')

[TOKEN$K_TICK_SUCC] = UPLIT (%ASCIC 'SUCC')

[TOKEN$K_TICK_VAL] = UPLIT (%ASCIC 'VAL')
    1478
                                 1606
1607
    1479
                                                                                                                                                                                                         - TABLEBASE,
   1480
1481
1482
1483
1484
1485
                                  1608
                                                                                                                                                                                                          - TABLEBASE.
                                  1609
                                                                                                                                                                                                          - TABLEBASE,
                                  1610
                                                                                                                                                                                                          - TABLEBASE
                                  1611
                                                                                                                                                                                                         - TABLEBASE,
                                 1612
                                                                                                                                                                                                         - TABLEBASE
                                                                                                                                                                                                         - TABLEBASE,
   1486
1487
1488
1489
                                 1614
                                                                                                                                                                                                         - TABLEBASE
                                 1615
                                                                                                                                                                                                          - TABLEBASE):
                            1616
1617
1618
P 1619
                                                      Define the ADA Terminator Lexical Token Table for subscript expressions.
   1491
1492
1493
1494
1496
1497
1498
1503
1504
1507
1508
1507
1513
1513
1514
1515
                                                 TERMINATOR_TABLE(ADA_SUBSCR_TERM_TBL,
TERMINATOR_ENTRY(')', TERM_CLOSE),
TERMINATOR_ENTRY(':', TERM_COLON, MUST_BE_SINGLE),
TERMINATOR_ENTRY('.', TERM_COLON),
TERMINATOR_ENTRY(',', TERM_COMMA));
                            P 1620
P 1621
P 1622
1623
1624
1625
                                 1626
1627
1628
1629
1630
1631
1633
1634
1635
1636
1637
1638
                                                  ! Define the ADA Predefined Identifier Table.
                                                  PRID_TABLE(ADA_PRID_TABLE);
                                                  ! Define the ADA Built-in Function Table.
                                                  BUILT_IN_FUNCTION_TABLE(ADA_FUNCTION_TABLE);
                                                      Define the ADA Number Scanner State Table. This table defines the states
                                                      of a finite-State Machine which picks up all valid numeric constants in the
                                                      language.
                                 1640
1641
1642
1643
1644
                                                      Each transition is of the form:
                                                                   NUMBER_TRANSITION(character-class, action-index, next-state)
                                                      The ADA standard defines a number to be of the form:
   1516
```

```
VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                         1702
1703
1704
1705
1706
1707
                                                                                          NUMBER_TRANSITION(OTHER, NOT_NUMBER, END_STATE)),
                                                                             NUMBER_STATE(GET_EXP_SIGN,
NUMBER_TRANSITION(DIGIT, DO_NOTHING, ACCUM_EXP),
NUMBER_TRANSITION(OTHER, NOT_NUMBER, END_STATE)),
                                                                             NUMBER_STATE(ACCUM_EXP,
NUMBER_TRANSITION(DIGIT, DO_NOTHING, ACCUM_EXP),
NUMBER_TRANSITION(UNDERSCORE, DO_NOTHING, T_ACCUM_EXP),
NUMBER_TRANSITION(OTHER, GOT_NUMBER, END_STATE)),
 1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
                                         1708
                                         1709
                                          1710
                                         1712
1713
1714
1715
                                                                             NUMBER_STATE(T_ACCUM_EXP,
NUMBER_TRANSITION(DIGIT, DO_NOTHING, ACCUM_EXP),
NUMBER_TRANSITION(OTHER, NOT_NUMBER, END_STATE)),
                                         1716
                                                                             NUMBER_STATE(B_START_STATE,
NUMBER_TRANSITION(DIGIT, DO_NOTHING, B_ACCUM_INT),
NUMBER_TRANSITION(OTHER, NOT_NUMBER, END_STATE)),
                                    P 1718
P 1719
                                   P 1720
P 1721
P 1722
P 1723
P 1724
P 1725
1592
                                                                            NUMBER_TRANSITION(DIGIT, DO_NOTHING, B_ACCUM_INT),
NUMBER_TRANSITION(HEXDIGIT, DO_NOTHING, B_ACCUM_INT),
NUMBER_TRANSITION(B, DO_NOTHING, B_ACCUM_INT),
NUMBER_TRANSITION(D, DO_NOTHING, B_ACCUM_INT),
NUMBER_TRANSITION(E, DO_NOTHING, B_ACCUM_INT),
NUMBER_TRANSITION(UNDERSCORE, DO_NOTHING, T_B_ACCUM_INT),
NUMBER_TRANSITION(DOT, MARK_DEC_PT, B_ACCUM_FRAC),
NUMBER_TRANSITION(POUND, DO_NOTHING, GET_EXPONENT),
NUMBER_TRANSITION(OTHER, GOT_NUMBER, END_STATE)),
1594
 1595
1596
 1597
                                   P 1726
P 1727
 1598
 1599
                                   P 1728
P 1729
 1600
 1601
                                    P 1730
 1602
                                   P 1731
P 1732
P 1733
 1603
                                                                             NUMBER_STATE(T_B_ACCUM_INT,
NUMBER_TRANSITION(DIGIT, DO_NOTHING, B_ACCUM_INT),
NUMBER_TRANSITION(HEXDIGIT, DO_NOTHING, B_ACCUM_INT),
 1604
1605
                                   P 1734
P 1735
 1606
                                                                                        NUMBER_TRANSITION(B, DO NOTHING, B ACCUM_INT),
NUMBER_TRANSITION(D, DO NOTHING, B ACCUM_INT),
NUMBER_TRANSITION(E, DO NOTHING, B ACCUM_INT),
NUMBER_TRANSITION(OTHER, NOT_NUMBER, END_STATE)),
1607
                                         1736
1737
                                    P
 1608
                                    P
1609
                                         1738
1739
                                    P
1610
                                    P
1611
                                                                           NUMBER STATE (B ACCUM FRAC,
NUMBER TRANSITION (DIGIT, DO NOTHING, B ACCUM FRAC),
NUMBER TRANSITION (HEXDIGIT, DO NOTHING, B ACCUM FRAC),
NUMBER TRANSITION (B, DO NOTHING, B ACCUM FRAC),
NUMBER TRANSITION (D, DO NOTHING, B ACCUM FRAC),
NUMBER TRANSITION (E, DO NOTHING, B ACCUM FRAC),
NUMBER TRANSITION (UNDERSCORE, DO NOTHING, T B ACCUM FRAC),
NUMBER TRANSITION (POUND, MARK E EXP, GET EXPONENT),
NUMBER TRANSITION (OTHER, NOT NUMBER, END STATE)),
1612
                                    P
                                         1740
                                    P
                                         1741
                                   P 1742
P 1743
1614
1615
                                    P 1744
1616
1617
                                    P
                                         1745
                                         1746
                                    P
 1618
1619
                                    P
1620
1621
1622
1623
1624
1625
                                    P
                                         1748
                                                                                          NUMBER_TRANSITION(OTHER, NOT_NUMBER, END_STATE)),
                                    P
                                         1749
                                         1750
1751
1752
1753
1754
1755
1756
1757
                                                                             NUMBER STATE(T B ACCUM FRAC,

NUMBER TRANSITION(DIGIT, DO NOTHING, B ACCUM FRAC),

NUMBER TRANSITION(HEXDIGIT, DO NOTHING, B ACCUM FRAC),

NUMBER TRANSITION(B, DO NOTHING, B ACCUM FRAC),

NUMBER TRANSITION(D, DO NOTHING, B ACCUM FRAC),

NUMBER TRANSITION(E, DO NOTHING, B ACCUM FRAC),

NUMBER TRANSITION(E, DO NOTHING, B ACCUM FRAC),

NUMBER TRANSITION(OTHER NOT NUMBER FND STATE))
                                    P
                                    P
                                    P
                                    P
 1626
1627
1628
1629
                                    P
                                    P
                                                                                          NUMBER_TRANSITION(OTHER, NOT_NUMBER, END_STATE)),
                                    P
1630
                                                                              NUMBER_STATE (END_STATE,
```

PRIMARY_STATE(END_STATE));

PRIMARY_STATE(GOT_SUPSCRIPT,
PRIMARY_TRANSITION(DOT, SUBSCR_DOT, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, SUBSCR_SUBSCR, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PRIMARY_TERM, SUBSCR_TERM, END_STATE)),

1811

1683

DBGPARSER V04-000	D 16 16-Sep- 14-Sep-	1984 02:10:13 1984 12:17:30	VAX-11 Bliss-32 V4.0-742 LDEBUG.SRCJDBGPARSER.B32;1
: 1688 : 1689 : 1690 : 1691 : 1692 : 1693 : 1694 : 1695 : 1696 : 1697 : 1698	1816 1 ! Define the table of pointers to the parse tab 1817 1 ! P 1818 1 LANGUAGE_TABLES(LANGUAGE = ADA, P 1819 1		

Page 32 (6)

P 1840 P 1841 P 1842 P 1844 P 1845 P 1846 P 1847 1848 1855 P 1855 P 1857 P 1857 P 1857

1860 1861 1862 1863 1864 P 1865 P 1866 P 1867 P 1870 P 1877 P 1873 P 1877 P 1877 P 1878 P 1879 P 1880

P 1881 P 1882 P 1883

BASIC PARSE TABLES

This section includes all the Lexical Scanner and Parser tables needed to scan and parse BASIC expressions.

Define the BASIC Character Table. What is listed here is actually a list of exceptions to the Character Table for language UNKNOWN. (Language UNKNOWN lists the "average" use of each character in the character set.)

```
CHAR_ENTRY('.', DOT, NUMBER_START, IDENT_MIDDLE, IDENT_END, ADDRESS_OP, SPECIAL_SYMBOL),
CHAR_ENTRY('%', OTHER, IDENT_END),
CHAR_ENTRY(':', OTHER, OPCHAR, OPCHAR_INFIX, TERMINATOR),
CHAR_ENTRY('^', OTHER, OPCHAR, SPECIAL_SYMBOL),
CHAR_ENTRY('<', OTHER, OPCHAR, OPCHAR_INFIX, ADDRESS_OP, TERMINATOR),
CHAR_ENTRY('>', OTHER, OPCHAR, OPCHAR_INFIX, TERMINATOR),
CHAR_ENTRY('>', OTHER, OPCHAR, OPCHAR_INFIX, TERMINATOR),
CHAR_ENTRY('=', OTHER, OPCHAR, OPCHAR_INFIX, TERMINATOR));
```

Define the BASIC Operator Table for operators whose names are identifiers.

```
OPERATOR_TABLE(BASIC_IDENT_OPTBL,

OPERATOR_ENTRY('NOT', BIT_NOT, PREFIX, 200, 45),

OPERATOR_ENTRY('AND', BIT_AND, INFIX, 40, 40),

OPERATOR_ENTRY('OR', BIT_OR, INFIX, 30, 30),

OPERATOR_ENTRY('XOR', BIT_XOR, INFIX, 30, 30),

OPERATOR_ENTRY('IMP', BIT_IMP, INFIX, 20, 20),

OPERATOR_ENTRY('EQV', BIT_EQV, INFIX, 10, 10));
```

Define the BASIC Operator Table for operators whose names are composed of operator characters such as "+", "-", or "*". This table includes those operators which are part of DEBUG Primary Symbols (such as "\").

```
OPERATOR_TABLE (BASIC_OPCHAR_OPTBL,
OPERATOR_ENTRY('\', GL
OPERATOR_ENTRY('\', BA
OPERATOR_ENTRY('\', DO
OPERATOR_ENTRY('\', SU
                                                                                                                                                                                                                                                                                                                                                                                        GLOBAL SLASH,
BACKSLASH,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      0000
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 0000
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     PREFIX,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   PRIMARY),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      INFIX,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   PRIMARY),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    INFIX,
POSTFIX,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    PRIMARY),
                                                                                                                                                                                                                                                                                                                                                                                           DOT
                                                                                                                                                                                                                                                                                                                                                                                           SUBSCRIPT,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   PRIMARY),
                                                                                            OPERATOR ENTRY('(',
OPERATOR ENTRY(')',
OPERATOR ENTRY('+',
OPERATOR ENTRY('+',
OPERATOR ENTRY('+',
OPERATOR ENTRY('**,
OPERATOR ENTRY('**,
OPERATOR ENTRY('**,
OPERATOR ENTRY('+',
OPERATOR ENTRY('+',
OPERATOR ENTRY('-',
OPERATOR ENTRY('-',
OPERATOR ENTRY('<-',
OPERATOR ENTRY('--',
OPERATOR ENTRY('---',
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 PREFIX, 200, POSTFIX, 6. PREFIX, 200, INFIX, 92, INFIX, 92, INFIX, 80, INFIX, 80, INFIX, 60, INFIX, 50, INFIX,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     200.
700.
700.
900.
800.
800.
600.
500.
                                                                                                                                                                                                                                                                                                                                                                                         OPENPAREN,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               LEXICAL),
                                                                                                                                                                                                                                                                                                                                                                                          CLOSEPAREN.
                                                                                                                                                                                                                                                                                                                                                                                         UNARY PLUS,
UNARY MINUS,
                                                                                                                                                                                                                                                                                                                                                                                        POWER OF,
POWER OF,
MULTIPLY,
                                                                                                                                                                                                                                                                                                                                                                                          DIVIDE.
                                                                                                                                                                                                                                                                                                                                                                                            ADD.
                                                                                                                                                                                                                                                                                                                                                                                        SUBTRACT,
LSS_THAN,
LSS_EQUAL,
                                                                                                                                                                                                                                                                                                                                                                                           LSS EQUAL,
```

```
F 16
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                                VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1
                                                    OPERATOR_ENTRY('>'
OPERATOR_ENTRY('>=',
OPERATOR_ENTRY('=>',
OPERATOR_ENTRY('=',
OPERATOR_ENTRY('<>',
OPERATOR_ENTRY('<>',
                          1884
1885
1886
1887
1888
1889
                                                                                            GTR_EQUAL,
GTR_EQUAL,
GTR_EQUAL,
EQUAL,
NOT_EQUAL,
NOT_EQUAL,
                                                                                                                      INFIX.
INFIX.
INFIX.
INFIX.
                                                                                                                                             50).
                                                                                                                                     50.
  INFIX,
                           1890
                      1892
1893
1894
1895
1896
P 1897
P 1898
P 1899
                                          Define the BASIC Terminator Lexical Token Table for subscript expressions. In BASIC a subscript expression can be terminated by ')' (end of sub-
                                          In BASIC a subscript expression can be terminated by ")" (end of subscripts), by "," (more subscripts to follow), or by ":" (string subscript upper bound to follow).
                                      TERMINATOR_TABLE(BASIC_SUBSCR_TERM_TBL,
TERMINATOR_ENTRY(')', TERM_CLOSE, BALANCED_PARENS),
TERMINATOR_ENTRY(':', TERM_COLON, MUST_BE_SINGLE),
TERMINATOR_ENTRY(',', TERM_COMMA));
                          1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
                                       ! Define the BASIC Predefined Identifier Table.
                                       PRID_TABLE(BASIC_PRID_TABLE);
                                       ! Define the BASIC Built-in Function Table.
                                       BUILT_IN_FUNCTION_TABLE(BASIC_FUNCTION_TABLE);
                          1912
                                          Define the BASIC Number Scanner State Table. This table defines the states
                          1914
                                          of a Finite-State Machine which picks up all valid numeric constants in the
                                          language.
                      The BASIC number table is the same as the UNKNOWN number table at present.
                                       BIND
                                              BASIC_NUMBER_TABLE = UNKNOWN_NUMBER_TABLE;
                                          Define the Primary Parser State Table for language BASIC Each Transition
                                          Entry in the state table has this format:
                                                    PRIMARY_TRANSITION(operator-code, action, next-state)
                                           where the first parameter is the operator code which causes the transition
                                           to be taken, the second parameter is the action routine CASE index for the
                                           transition, and the third parameter is the next state in the finite-State
                                          Machine.
                                       PRIMARY_STATE_TABLE(BASIC_PRIMARY_TABLE,
                                             PRIMARY_TRANSITION(GLOBAL_SLASH, START_GBL, GET_GLOBAL),
PRIMARY_TRANSITION(BACKSLASH, START_SLASH, GOT_BACKSLASH),
PRIMARY_TRANSITION(INVOCNUM, SLASH_INVOCNUM, GOT_BACKSLASH),
PRIMARY_TRANSITION(DOT, START_DOT, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, START_SUBSCR, GOT_SUBSCRIPT),
                       P 1940
```

Page

BACKSLASH,

1908

BIF_TABLE = BLISS_FUNCTION_TABLE);

(9)

This section includes all the Lexical Scanner and Parser tables needed to scan and parse the C language.

Define the C Character Table. What is listed here is actually a list of exceptions to the Character Table for Language UNKNOWN.

```
CHAR_EXCEPTION_TABLE(C_CHARTBL,

CHAR_ENTRY('$', OTHER,

CHAR_ENTRY('', OTHER,

CHAR_ENTRY('*', OTHER,

CHAR_ENTRY('*', OTHER,

CHAR_ENTRY('*', OTHER,

CHAR_ENTRY('*', OTHER,

CHAR_ENTRY(''', OTHER,

CHAR_ENTRY(''', OTHER,

CHAR_ENTRY(''', OTHER,

CHAR_ENTRY(''', OTHER,

CHAR_ENTRY('''', OTHER,

CHAR_ENTRY('''', OTHER,

CHAR_ENTRY('''', OTHER,

CHAR_ENTRY('''', OTHER,

CHAR_ENTRY(''''', OTHER,

CHAR_ENTRY(''''', OTHER,

CHAR_ENTRY(''''', OTHER,

CHAR_ENTRY('''''', OTHER,
                                                                                                                                                                                                    IDENT_ANYWHERE),
IDENT_ANYWHERE),
OPCHAR, OPCHAR_INFIX),
OPCHAR, ADDRESS_OP),
OPCHAR),
                                                                                                                                                                                                    OPCHAR, OPCHAR_INFIX, ADDRESS_OP),
OPCHAR, OPCHAR_INFIX, TERMINATOR),
OPCHAR, OPCHAR_INFIX, TERMINATOR),
OPCHAR),
TERMINATOR),
                                                                                                                                                                                                     OPCHAR, SPECIAL SYMBOL),
OPCHAR, OPCHAR INFIX),
                                                                                                                                                                                                     OPCHAR));
```

Define the C Operator Table for operators whose names are identifiers.

```
OPERATOR_TABLE (C_IDENT_OPTBL
        OPERATOR_ENTRYT'SIZEOF', SIZEOF,
                                                 PREFIX, 200, 140));
```

Define the C Operator Table for operators whose names are composed of operator characters such as "+", "-", or "*". This table includes operators which are part of DEBUG Primary Symbols (such as "\").

```
OPERATOR TABLE (C_OPCHAR_OPTBL,
OPERATOR ENTRY('\',
OPERATOR ENTRY('\',
OPERATOR ENTRY('\',
                                                                                                                                                                                                                                                                                                                                                                                        GLOBAL SLASH,
BACKSLASH,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     PREFIX. 0. 0. PRIMARY).
INFIX. 0. 0. PRIMARY).
POSTFIX.0. 0. PRIMARY).
INFIX. 0. 0. PRIMARY).
                                                                                                                                                                                                                                                                                                                                                                                           SUBSCRIPT,
                                                                                                OPERATOR_ENTRY('.';
                                                                                                                                                                                                                                                                                                                                                                                          DOT.
                                                                                           OPERATOR ENTRY('(',
OPERATOR ENTRY(')',
OPERATOR ENTRY('!',
OPERATOR ENTRY(''',
OPERATOR ENTRY('*',
OPERATOR ENTRY('*',
OPERATOR ENTRY('*',
OPERATOR ENTRY(''',
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 PREFIX, 200, POSTFIX, 6, PREFIX, 200, PREFIX, 200, INFIX, 130, INFIX, 130, INFIX, 130, INFIX, 110, INFIX, 110, INFIX, 100, INFIX, INF
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     5, LEXICAL),
200, LEXICAL),
140),
140),
140),
130),
130),
130),
                                                                                                                                                                                                                                                                                                                                                                                          OPENPAREN,
                                                                                                                                                                                                                                                                                                                                                                                           CLOSEPAREN.
                                                                                                                                                                                                                                                                                                                                                                                         NOT,
BIT_NOT
                                                                                                                                                                                                                                                                                                                                                                                           INDIRECT,
                                                                                                                                                                                                                                                                                                                                                                                           MULTIPLY,
                                                                                                                                                                                                                                                                                                                                                                                          DIVIDE.
                                                                                                                                                                                                                                                                                                                                                                                        REMAINDER,
LEFT_SHIFT,
RIGHT_SHIFT,
LSS_TRAN,
LSS_EQUAL,
GTR_THAN,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           110).
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          110),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      100,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         100).
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               100).
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         INFIX.
```

```
DBGPARSER
V04-000
                                                                                                                                        VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                                                                                                                                                                                 Page
                                                 OPERATOR_ENTRY('>=',
OPERATOR_ENTRY('==',
OPERATOR_ENTRY('!=',
OPERATOR_ENTRY(':=',
OPERATOR_ENTRY(':-',
OPERATOR_ENTRY(':-',
                                                                                      GTR EQUAL,
EQUAL,
NOT EQUAL,
BIT XOR,
BIT OR,
SHORT OR,
                                                                                                                                   100).
90).
90).
70).
60).
40));
                                                                                                                INFIX.
INFIX.
INFIX.
INFIX.
                                                                                                                            100.
90.
70.
60.
40.
                        222
                                                                                                                INFIX.
                                                                                                                INFIX.
                                        Define Lexical Token Entries which require special-case scanning.
                                     BIND
                                           C_ADDR OF TOKEN =
OPERATOR ENTRY('&',
C_BIT_AND_TOKEN =
                                                                                       ADDRESS_OF,
                                                                                                                PREFIX, 200, 140).
                                           OPERATOR ENTRY('&', C_AND_TOKEN =
                                                                                       BIT_AND,
                                                                                                                INFIX.
                                                                                                                              80, 80),
                                           OPERATOR_ENTRY('&&', C_ADD_TOKEN =
                                                                                       SHORT_AND,
                                                                                                                INFIX.
                                                                                                                              50, 50),
                                                 OPERATOR_ENTRY('+',
                                                                                       ADD.
                                                                                                                INFIX, 120, 120),
                                           C_MINUS_TOKEN =
                                                 OPERATOR_ENTRY('-',
                                                                                       UNARY_MINUS.
                                                                                                                PREFIX, 200, 140).
                                           C_SUB_TOKEN =
                                                 OPERATOR_ENTRY('-',
                                                                                       SUBTRACT.
                                                                                                                INFIX, 120, 120),
                                           C_ARROW_TOKEN =
                                                 OPERATOR_ENTRY("->".
                                                                                       PLI_DEREF,
                                                                                                                INFIX.
                                                                                                                                       O, PRIMARY),
                                              The indirect operator will also be allowed as an address expression operator in ( (synonymous with "." and "a"). That's why it appears here as a special case, in addition to its appearance in the normal operator tables. The precedence of "40" here is relative to other
                        2209
2210
2211
2212
2213
2214
2215
2216
22219
22221
22221
22221
22223
                                              address expression operators.
                                           C INDIRECT TOKEN =
                                                 OPERATOR_ENTRY('*'.
                                                                                       INDIRECT,
                                                                                                               PREFIX, 200, 40);
                                        Increment and decrement were commented out so as not to allow operators with
                                        side effects. If we decide to allow them, the comments can be removed.
                                     !BIND
                                             C_PRE_INCR_TOKEN =
                                                 OPERATOR_ENTRY("++",
                                                                                      PRE_INCR,
                                                                                                               PREFIX, 200, 140),
                                             C_POST_INCR_TOKEN =
                                             OPERATOR ENTRY ( '++', C PRE DECR TOKEN =
                                                                                      POST_INCR,
                                                                                                               POSTFIX, 140, 200),
                                                 OPERATOR_ENTRY('--',
                                                                                       PRE_DECR,
                                                                                                               PREFIX, 200, 140),
                                             C_POST_DECR_TOKEN =
                                                 OPERATOR_ENTRY('--',
                                                                                       POST_DECR,
                                                                                                               POSTFIX,140, 200);
                                        Define the C Terminator Lexical Token Table for subscript expressions.
                                     TERMINATOR TABLE(C_SUBSCR_TERM_TBL,
TERMINATOR_ENTRY(']', TERM_CLOSE),
TERMINATOR_ENTRY(':', TERM_COLON, MUST_BE_SINGLE),
TERMINATOR_ENTRY(',', TERM_COMMA));
```

```
999
```

```
Define the C Predefined Identifier Table.
PRID_TABLE(C_PRID_TABLE);
   Define the C Built-in Function Table.
BUILT_IN_FUNCTION_TABLE(C_FUNCTION_TABLE);
   Define the C Number Scanner State Table. This table defines the states
   of a Finite-State Machine which picks up all valid numeric constants in the
   language.
   The C number table is the same as the UNKNOWN number table.
       C_NUMBER_TABLE = UNKNOWN_NUMBER_TABLE;
   Define the Primary Parser State Table for language C. Each transition
   Entry in the state table has this format:
             PRIMARY_TRANSITION(operator-code, action, next-state)
   where the first parameter is the operator code which causes the transition
   to be taken, the second parameter is the action routine CASE index for the
   transition, and the third parameter is the next state in the Finite-State
   Machine.
PRIMARY_STATE_TABLE(C_PRIMARY_TABLE,
     PRIMARY STATE (START STATE,

RIMARY TRANSITION (GLOBAL SLASH, START GBL, GET GLOBAL),

RIMARY TRANSITION (BACKSLASH, START SLASH, GOT BACKSLASH),

PRIMARY TRANSITION (INVOCNUM, SLASH INVOCNUM, GOT BACKSLASH),

PRIMARY TRANSITION (DOT, START DOT, GOT DOT),

PRIMARY TRANSITION (SUBSCRIPT, START SUBSCR, GOT SUBSCRIPT),

PRIMARY TRANSITION (PLI DEREF, START DEREF, GOT DOT),

PRIMARY TRANSITION (PRIMARY TERM, START TERM, END_STATE)),
      PRIMARY_STATE(GET_GLOBAL,
PRIMARY_TRANSITION(PRIMARY_TERM, GBL_TERM, END_STATE)),
      PRIMARY STATE (GOT BACKSLASH,
PRIMARY TRANSITION (BACKSLASH, SLASH SLASH, GOT BACKSLASH),
PRIMARY TRANSITION (INVOCNUM, SLASH INVOCNUM, GOT BACKSLASH),
PRIMARY TRANSITION (DOT, SLASH DOT, GOT DOT),
PRIMARY TRANSITION (SUBSCRIPT, SLASH SUBSCR, GOT SUBSCRIPT),
PRIMARY TRANSITION (PLI DEREF, SLASH DEREF, GOT DOT),
PRIMARY TRANSITION (PRIMARY TERM, SLASH TERM, END STATE)),
      PRIMARY_STATE(GOT_DOT, PRIMARY_TRANSITION(DOT, DOT_DOT, GOT_DOT), PRIMARY_TRANSITION(SUBSCRIPT, DOT_SUBSCR, GOT_SUBSCRIPT),
```

DBO

Page

.....

Page 43 (10)

```
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                                                                                                                                                                                  VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1
                                                                                                                                                                                                                                                                                                                                                                                                                         Page
                                                                                                                                                                                                                                                                                                                                                                                                                                     (10)
                                                                                                        OPERATOR ENTRY('>',
OPERATOR ENTRY('<',
OPERATOR ENTRY('<',
OPERATOR ENTRY(''<',
OPERATOR ENTRY(')',
OPERATOR ENTRY(')',
OPERATOR ENTRY('*',
OPERA
                                                                                                                                                                                                                                            PREFIX,
PREFIX,
PREFIX,
PREFIX,
INFIX,
                                                                                                                                                                                                                                                                                       30, LEXICAL),
30, LEXICAL),
30, LEXICAL),
5, LEXICAL),
200, LEXICAL),
70),
60),
60),
50),
                                                                                                                                                                                        PREFIX_GTR,
PREFIX_LSS,
PREFIX_EQL,
                                                   OPENPAREN.
                                              P
                                                                                                                                                                                         CLOSEPAREN.
                                                                                                                                                                                        POWER OF,
MULTIPLY,
                                              P
                                              P
                                                                                                                                                                                        DIVIDE,
UNARY_PLUS,
UNARY_MINUS,
                                              P
                                              P
                                              P
                                              P
                                                                                                                                                                                         ADD.
                                              P
                                                                                                                                                                                         SUBTRACT.
                                                                                                                                                                                         GTR_THAN,
LSS_THAN,
                                              P
                                                                                                                                                                                         EQUAL,
                                                                                    Define the COBOL Terminator Lexical Token Table for subscript expressions.
                                                                              TERMINATOR_TABLE(COBOL_SUBSCR_TERM_TBL,
TERMINATOR_ENTRY(')', TERM_CLOSE),
TERMINATOR_ENTRY(':', TERM_COLON, MUST_BE_SINGLE),
TERMINATOR_ENTRY(',', TERM_COMMA));
                                              P
                                                                                    Define the COBOL Predefined Identifier Table.
                                                                               PRID_TABLE(COBOL_PRID_TABLE);
                                                                                    Define the COBOL Built-in Function Table.
                                                                               BUILT_IN_FUNCTION_TABLE(COBOL_FUNCTION_TABLE);
                                                                                     Define the COBOL Number Scanner State Table. This table defines the states
                                                                                      of a Finite-State Machine which picks up all valid numeric constants in the
                                                                                      language. Each Transition Entry is of the form:
                                                                                                          NUMBER_TRANSITION(character-class, action-index, next-state)
                                                                                      where the character-class and action-index names are automatically prefixed
                                                                                      by 'NUMST$K_CLASS_" or 'NUMST$K_ACT_" by the NUMBER_TRANSITION macro.
                                                                               NUMBER_STATE_TABLE(COBOL_NUMBER_TABLE,
                                              P
                                                                                            NUMBER_STATE(START_STATE,
NUMBER_TRANSITION(DIGIT, GO_PAST_PACK, ACCUM_INT),
NUMBER_TRANSITION(DOT, DO_NOTHING, LEADING_DOT),
NUMBER_TRANSITION(OTHER, NOT_NUMBER, END_STATE)),
                                              PPP
                                              P
                                              P
                                                                                            NUMBER_STATE(LEADING_DOT,
NUMBER_TRANSITION(DIGIT, GO_PAST_PACK_FRAC, ACCUM_FRAC),
NUMBER_TRANSITION(OTHER, NOT_NUMBER, END_STATE)),
                                              P
                                              P
                                              P
                                              PPP
                                                                                             NUMBER_STATE (ACCUM_INT
                                                                                                          NUMBER_TRANSITION(DIGIT, GO_PAST_PACK, ACCUM_INT),
```

VC

(10)

Page

```
DBGPARSER
V04-000
                                                                                                                                                               16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                                                                          VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32:1
                                                                               PRIMARY_TRANSITION(INVOCNUM, SLASH_INVOCNUM, GOT_BACKSLASH),
PRIMARY_TRANSITION(DOT, START_DOT_COB, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, START_SUBSCR_PLI, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PRIMARY_TERM, START_TERM, END_STATE)),
                                      PRIMARY_STATE(GET_GLOBAL, PRIMARY_TRANSITION(PRIMARY_TERM, GBL_TERM, END_STATE)),
                                                                    PRIMARY_STATE(GOT_BACKSLASH,
PRIMARY_TRANSITION(BACKSLASH, SLASH_SLASH, GOT_BACKSLASH),
PRIMARY_TRANSITION(INVOCNUM, SLASH_INVOCNUM, GOT_BACKSLASH),
PRIMARY_TRANSITION(SUBSCRIPT, SLASH_SUBSCR_PLI, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PRIMARY_TERM, SLASH_TERM, END_STATE)),
                                                                     PRIMARY_STATE(GOT_DOT,
PRIMARY_TRANSITION(BACKSLASH, DOT_SLASH_COB, GOT_BACKSLASH),
PRIMARY_TRANSITION(DOT, DOT_DOT_COB, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, DOT_SUBSCR_COB, GOT_SUBSCRIPT),
                                                                               PRIMARY_TRANSITION(PRIMARY_TERM, DOT_TERM_COB, END_STATE)),
                                                                     PRIMARY_STATE(GOT_SUBSCRIPT, PRIMARY_TRANSITION(SUBSCRIPT, SUBSCR_SUBSCR_PLI, GOT_SUBSCRIPT2), PRIMARY_TRANSITION(PRIMARY_TERM, SUBSCR_TERM_PLI, END_STATE)),
                                                                     PRIMARY_STATE(GOT_SUBSCRIPT2, PRIMARY_TRANSITION(PRIMARY_TERM, SUBSCR_TERM_PLI, END_STATE)),
                                                                      PRIMARY_STATE(END_STATE));
                                                                Define the table of pointers to the parse tables for COBOL.
                                                        LANGUAGE_TABLES(LANGUAGE = COBOL,
CHARTBL = COBOL CHARTBL,
IDENT OPTBL = COBOL IDENT OPTBL,
OPCHAR OPTBL = COBOL OPCHAR OPTBL,
NUMBER TABLE = COBOL NUMBER TABLE,
PRIMARY TABLE = COBOL PRIMARY TABLE,
SUBSCR TERMS = COBOL SUBSCR TERM TBL,
PRIDTBE = COBOL PRID TABLE,
BIF TABLE = COBOL FUNCTION TABLE,
MULTIPLE SUBSCR = FALSE,
COMPONENTS_IN_PATHNAME = TRUE);
                                  22222
```

Page 46 (10)

FORTRAN PARSE TABLES

This section includes all the Lexical Scanner and Parser tables needed to scan and parse FORTRAN expressions.

Define the FORTRAN Character Table. What is listed here is actually a list of exceptions to the Character Table for language UNKNOWN. (Language UNKNOWN lists the "average" use of each character in the character set.)

CHAR_EXCEPTION_TABLE(FORTRAN_CHARTBL, CHAR_ENTRY('.', DOT, NUMBER_START, SPECIAL_SYMBOL), CHAR_ENTRY('', OTHER, OPCHAR_INFIX));

Define the FORTRAN Operator Table for operators whose names are identifiers. This table is empty since FORTRAN has no such operators, but the Lexical Scanner requires that such a table exist anyway.

OPERATOR_TABLE (FORTRAN_IDENT_OPTBL);

Define the FORTRAN Operator Table for operators whose names are composed of operator characters such as "+", "-", or "*". This table includes those operators which are part of DEBUG Primary Symbols (such as "\").

```
OPERATOR_TABLE (FORTRAN_OPCHAR_OPTBL,

OPERATOR_ENTRY('\', GLOB
OPERATOR_ENTRY('\', BACK
OPERATOR_ENTRY('\', SUBS
OPERATOR_ENTRY('\', ADD,
OPERATOR_ENTRY('\', ADD,
OPERATOR_ENTRY('\', SUBS
OPERATOR_ENTRY('\', UNAF
OPERATOR_ENTRY('\', UNAF
OPERATOR_ENTRY('\', MULI
OPERATOR_ENTRY('\', DIVI
OPERATOR_ENTRY('\', DIVI
OPERATOR_ENTRY('\', OPERATOR_ENT
                                                                                                                                                                                                                                                                                                                                 GLOBAL SLASH,
BACKSLASH,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             PREFIX,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  0.
0.
60).
70).
80).
80).
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          PRIMARY),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          PRIMARY),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              INFIX,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          PRIMARY),
                                                                                                                                                                                                                                                                                                                                  SUBSCRIPT,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              POSTFIX,
                                                                                                                                                                                                                                                                                                                                 DOT,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                INFIX.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          PRIMARY),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                INFIX,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           INFIX,
PREFIX,
PREFIX,
INFIX,
INFIX,
                                                                                                                                                                                                                                                                                                                                  SUBTRACT,
                                                                                                                                                                                                                                                                                                                                UNARY_PLUS,
UNARY_MINUS,
MULTIPLY,
                                                                                                                                                                                                                                                                                                                                 DIVIDE
                                                                                                                                                                                                                                                                                                                                 POWER OF,
CONCATENATE,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                INFIX.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         INFIX.
PREFIX.
POSTFIX.
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     200.
                                                                                                                                                                                                                                                                                                                                  OPENPAREN,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         200. LEXICAL));
                                                                                                                                                                                                                                                                                                                                  CLOSEPAREN,
```

BIND

FORTRAN_INDIRECT_TOKEN = OPERATOR_ENTRY('.', INDIRECT, PREFIX, 200, 40), FORTRAN_DOT_TOKEN = OPERATOR_ENTRY('.', DOT, INFIX, 0, 0, PRIMARY);

Define an Operator Table for FORTRAN operators of the form .XX. or .XXX. This includes all the FORTRAN comparison and boolean operators.

OPERATOR_TABLE(FORTRAN_SPECIAL_OPTBL,
OPERATOR_ENTRY('.eq.', EQUAL, INFIX, 50, 50)
OPERATOR_ENTRY('.NE.', NOT_EQUAL, INFIX, 50, 50)

```
H 1
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                              VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGPARSER.B32;1
                                                                                                                                                                                                              (11)
                                                                                                                               50, 50),
50, 50),
50, 50),
50, 50),
200, 40),
30, 30),
20, 20),
10, 10),
10, 10),
                                                   OPERATOR_ENTRY('.GT.',
OPERATOR_ENTRY('.GE.',
OPERATOR_ENTRY('.LT.',
OPERATOR_ENTRY('.LE.',
OPERATOR_ENTRY('.NOT.',
OPERATOR_ENTRY('.AND.',
OPERATOR_ENTRY('.OR.',
OPERATOR_ENTRY('.XOR.',
OPERATOR_ENTRY('.EQV.',
OPERATOR_ENTRY('.NEQV.',
                                                                                                                    INFIX,
INFIX,
INFIX,
PREFIX,
INFIX,
                                                                                          GTR_THAN,
GTR_EQUAL,
LSS_THAN,
LSS_EQUAL,
   NOT,
                                                                                          AND,
                                                                                                                     INFIX,
                                                                                          XOR,
                                                                                                                     INFIX.
                                                                                          EQV.
                                                                                                                     INFIX,
                                                                                                                    INFIX.
                                         Define the FORTRAN Terminator Lexical Token Table for subscript expressions. In FORTRAN, a subscript expression can be terminated by ")" (end of subscripts), by "," (more subscripts to follow), or by ":" (string subscript upper bound to follow).
                                      TERMINATOR_TABLE(FORTRAN_SUBSCR_TERM_TBL,
TERMINATOR_ENTRY(')', TERM_CCOSE, BALANCED_PARENS),
TERMINATOR_ENTRY('.', TERM_COMMA),
TERMINATOR_ENTRY(':', TERM_COLON));
                                         Define the FORTRAN Predefined Identifier Table.
                                      PRID_TABLE(FORTRAN_PRID_TABLE,
PRID_ENTRY('.TRUE.', ATOMIC, L, 1),
PRID_ENTRY('.FALSE.', ATOMIC, L, 0));
                                         Define the FORTRAN Built-in Function Table.
                                      BUILT_IN_FUNCTION_TABLE(FORTRAN_FUNCTION_TABLE);
                                         Define the FORTRAN Number Scanner State Table. This table defines the states
                                          of a Finite-State Machine which picks up all valid numeric constants in the
                                          language.
                                          The FORTRAN number table is the same as the number table for language UNKNOWN.
                                             FORTRAN_NUMBER_TABLE = UNKNOWN_NUMBER_TABLE;
                                          Define the Primary Parser State Table for Language FORTRAN. Each Transition
                                          Entry in the state table has this format:
                                                    PRIMARY_TRANSITION(operator-code, action, next-state)
                                          where the first parameter is the operator code which causes the transition
                                          to be taken, the second parameter is the action routine CASE index for the
                                          transition, and the third parameter is the next state in the finite-State
                                          Machine.
                                       PRIMARY_STATE_TABLE (FORTRAN_PRIMARY_TABLE,
```

BIF_TABLE = FORTRAN_FUNCTION_TABLE);

Page

(11)

0000000

MACRO PARSE TABLES

This section includes all the Lexical Scanner and Parser tables needed to scan and parse the MACRO language.

Define the MACRO Character Table. What is listed here is actually a list of exceptions to the Character Table for Language UNKNOWN.

```
CHAR_ENTRY('', OTHER, IDENT_ANYWHERE),
CHAR_ENTRY('S', OTHER, IDENT_ANYWHERE),
CHAR_ENTRY('S', OTHER, IDENT_ANYWHERE),
CHAR_ENTRY('.', DOT, NUMBER_START, OPCHAR, ADDRESS_OP, SPECIAL_SYMBOL,
IDENT_MIDDLE, IDENT_END),
CHAR_ENTRY('<', OTHER, OPCHAR, ADDRESS_OP),
CHAR_ENTRY('>', OTHER, TERMINATOR));
```

Define the MACRO Operator Table for operators whose names are identifiers.

```
OPERATOR_ENTRY('EQU', EQUAL, INFIX, 50, 50),
OPERATOR_ENTRY('EQU', EQUAL, INFIX, 50, 50),
OPERATOR_ENTRY('NEQU', NOT_EQUAL, INFIX, 50, 50),
OPERATOR_ENTRY('NEQU', NOT_EQUAL, INFIX, 50, 50),
OPERATOR_ENTRY('NEQU', NOT_EQUAL, INFIX, 50, 50),
OPERATOR_ENTRY('GTR', GTR_THAN, INFIX, 50, 50),
OPERATOR_ENTRY('GTRU', GTR_THAN, INFIX, 50, 50),
OPERATOR_ENTRY('GEQU', GTR_EQUAL, INFIX, 50, 50),
OPERATOR_ENTRY('GEQU', GTR_EQUAL, INFIX, 50, 50),
OPERATOR_ENTRY('LSS', LSS_THAN, INFIX, 50, 50),
OPERATOR_ENTRY('LSSU', LSS_THAN, INFIX, 50, 50),
OPERATOR_ENTRY('LSSU', LSS_THAN, INFIX, 50, 50),
OPERATOR_ENTRY('LEQU', LSS_EQUAL, INFIX, 50, 50),
OPERATOR_ENTRY('LEQU', LSS_EQUAL, INFIX, 50, 50),
OPERATOR_ENTRY('NOT', BIT_NOT, PREFIX, 200, 40),
OPERATOR_ENTRY('NOT', BIT_AND, INFIX, 30, 30),
OPERATOR_ENTRY('NOT', BIT_AND, INFIX, 30, 30),
OPERATOR_ENTRY('OR', BIT_EQV, INFIX, 10, 10),
OPERATOR_ENTRY('XOR', BIT_XOR, INFIX, 10, 10),
OPERATOR_ENTRY('MOD', REMAINDER, INFIX, 70, 70));
```

Define the MACRO Operator Table for operators whose names are composed of operator characters such as "+", "-", or "*". This table includes operators which are part of DEBUG Primary Symbols (such as "\").

```
OPERATOR_ENTRY('\', GLOBAL_SLASH, PREFIX, 0, 0, PRIMARY),
OPERATOR_ENTRY('\', BACKSLASH, INFIX, 0, 0, PRIMARY),
OPERATOR_ENTRY('\', OPENPAREN, OPERATOR_ENTRY('\', CLOSEPAREN, OPERATOR_ENTRY('\', BITSELECT, OPERATOR_ENTRY('\', BITSELECT, OPERATOR_ENTRY('\', INDIRECT, OPERATOR_
```

Page 52

PASCAL PARSE TABLES

This section includes all the Lexical Scanner and Parser tables needed to scan and parse the PASCAL language.

Define the PASCAL Character Table. What is listed here is actually a list of exceptions to the Character Table for Language UNKNOWN.

```
CHAR_ENTRY('<', OTHER, OPCHAR, OPCHAR_INFIX, ADDRESS_OP),
CHAR_ENTRY('=', OTHER, OPCHAR, OPCHAR_INFIX, TERMINATOR),
CHAR_ENTRY('>', OTHER, OPCHAR, OPCHAR_INFIX, TERMINATOR),
CHAR_ENTRY('^', OTHER, OPCHAR, SPECIAL_SYMBOL),
CHAR_ENTRY('.', DOT, NUMBER_START, OPCHAR, ADDRESS_OP, SPECIAL_SYMBOL, TERMINATOR),
CHAR_ENTRY('[', OTHER, OPCHAR],
CHAR_ENTRY('[', OTHER, TERMINATOR));
```

! Define the PASCAL Operator Table for operators whose names are identifiers.

Define the PASCAL Operator Table for operators whose names are composed of operator characters such as "+", "-", or "*". This table includes operators which are part of DEBUG Primary Symbols (such as "\").

```
OPERATOR TABLE (PASCAL OPCHAR_OPTBL,

OPERATOR ENTRY('\'. GLOBAL SLASH,
OPERATOR ENTRY('\'. BACKSLÄSH,
OPERATOR ENTRY('\'. SUBSCRIPT,
OPERATOR ENTRY('\'. DOT,
OPERATOR ENTRY('\'. PASCAL DEREF,
OPERATOR ENTRY('\'. BIF_OP,
OPERATOR ENTRY('\'. BIF_OP,
OPERATOR ENTRY('\'. OPENPAREN,
OPERATOR ENTRY('\'. OPENPAREN,
OPERATOR ENTRY('\'. OPENSET,
OPERATOR ENTRY('\'. OPENSET,
OPERATOR ENTRY('\'. UNARY PLUS,
OPERATOR ENTRY('\'. UNARY PLUS,
OPERATOR ENTRY('\'. UNARY PLUS,
OPERATOR ENTRY('\'. UNARY PLUS,
OPERATOR ENTRY('\'. UNARY MINUS,
OPERATOR ENTRY('\'. MULTIPLY,
OPERATOR ENTRY('\'. MULTIPLY,
OPERATOR ENTRY('\'. MULTIPLY,
OPERATOR ENTRY('\'. DIVIDE,
OPERATOR ENTRY('\'. ADD,
OPERATOR ENTRY('\'. SUBTRACT,
OPERATOR ENTRY('\'. LSS_THAN,
OPERATOR ENTRY(\'\'. LSS_THAN,
O
```

```
N 1
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                         VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1
                                                 OPERATOR_ENTRY('<=',
OPERATOR_ENTRY('>',
OPERATOR_ENTRY('>=',
OPERATOR_ENTRY('=',
OPERATOR_ENTRY('<>',
                                                                                       LSS_EQUAL,
GTR_THAN,
GTR_EQUAL,
                                                                                                                INFIX.
INFIX.
INFIX.
                                                                                                                                      50),
50),
50),
50);
                                                                                                                              50.
  P
                     P
                                                                                                                 INFIX.
                                                                                        EQUAL .
                                                                                        NOT_EQUAL,
                                                                                                                 INFIX.
                                        Define the PASCAL Terminator Lexical Token Table for subscript expressions.
                                     TERMINATOR_TABLE(PASCAL_SUBSCR_TERM_TBL,
TERMINATOR_ENTRY(']', TERM_CLOSÉ),
TERMINATOR_ENTRY(':', TERM_COLON, MUST_BE_SINGLE),
TERMINATOR_ENTRY(',', TERM_COMMA));
                     P
                                        Define the PASCAL Predefined Identifier Table.
                                     PRID_TABLE(PASCAL_PRID_TABLE,
PRID_ENTRY('TRUE', ATOMIC, TF, 1),
PRID_ENTRY('FALSE', ATOMIC, TF, 0),
PRID_ENTRY('NIL', TPTR, Z, 0));
                     PP
                                        Define the PASCAL Built-in Function Table.
                                     BUILT_IN_FUNCTION_TABLE(PASCAL_FUNCTION_TABLE);
                                        Define the PASCAL Number Scanner State Table. This table defines the states
                                         of a Finite-State Machine which picks up all valid numeric constants in the
                                         language.
                                        The PASCAL number table is the same as the number table for language UNKNOWN.
                                     BIND
                                           PASCAL_NUMBER_TABLE = UNKNOWN_NUMBER_TABLE;
                                        Define the Primary Parser State Table for language PASCAL. Each transition
                                        Entry in the state table has this format:
                                                  PRIMARY_TRANSITION(operator-code, action, next-state)
                                         where the first parameter is the operator code which causes the transition
                                         to be taken, the second parameter is the action routine CASE index for the
                                         transition, and the third parameter is the next state in the Finite-State
                                        Machine.
                                      PRIMARY_STATE_TABLE (PASCAL_PRIMARY_TABLE,
                      000000000
                                           PRIMARY STATE (START STATE,
PRIMARY TRANSITION (GLOBAL SLASH, START GBL, GET GLOBAL),
PRIMARY TRANSITION (BACKSLASH, START SLASH, GOT BACKSLASH),
PRIMARY TRANSITION (INVOCNUM, SLASH INVOCNUM, GOT BACKSLASH),
PRIMARY TRANSITION (DOT, START DOT, GOT DOT),
PRIMARY TRANSITION (SUBSCRIPT, START SUBSCR, GOT SUBSCRIPT),
PRIMARY TRANSITION (PASCAL DEREF, START DEREF, GOT DEREF),
```

(13)

Page

```
DBGPARSER
                                                                                                                                                                                                       16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                                                                                                                                  VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
V04-000
                                                                                                   PRIMARY_TRANSITION(BIF OP, START_BIF CALL, END STATE), PRIMARY_TRANSITION(PRIMARY_TERM, START_TERM, END_STATE)),
                                                 PRIMARY_STATE(GET_GLOBAL, PRIMARY_TRANSITION(PRIMARY_TERM, GBL_TERM, END_STATE)),
                                                                                     PRIMARY STATE (GOT BACKSLASH,
PRIMARY TRANSITION (BACKSLASH, SLASH SLASH, GOT BACKSLASH),
PRIMARY TRANSITION (INVOCNUM, SLASH INVOCNUM, GOT BACKSLASH),
PRIMARY TRANSITION (DOT, SLASH DOT, GOT DOT),
PRIMARY TRANSITION (SUBSCRIPT, SLASH SUBSCR, GOT SUBSCRIPT),
PRIMARY TRANSITION (PASCAL DEREF, SLASH DEREF, GOT DEREF),
PRIMARY TRANSITION (PRIMARY TERM, SLASH TERM, END STATE)),
                                                                                     PRIMARY_STATE(GOT_DOT,
PRIMARY_TRANSITION(DOT, DOT_DOT, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, DOT_SUBSCR, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PASCAL_DEREF, DOT_DEREF, GOT_DEREF),
PRIMARY_TRANSITION(PRIMARY_TERM, DOT_TERM, END_STATE)),
                                                                                     PRIMARY_STATE(GOT_SUBSCRIPT,
PRIMARY_TRANSITION(DOT, SUBSCR_DOT, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, SUBSCR_SUBSCR, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PASCAL_DEREF, SUBSCR_DEREF, GOT_DEREF),
PRIMARY_TRANSITION(PRIMARY_TERM, SUBSCR_TERM, END_STATE)),
                                                                                     PRIMARY_STATE(GOT_DEREF,
PRIMARY_TRANSITION(DOT, DEREF_DOT, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, DEREF_SUBSCR, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PASCAL_DEREF, DEREF_DEREF, GOT_DEREF),
PRIMARY_TRANSITION(PRIMARY_TERM, DEREF_TERM, END_STATE)),
                                                                                       PRIMARY_STATE(END_STATE));
                                                                               Define the table of pointers to the parse tables for PASCAL.
                                                                        LANGUAGE_TABLES(LANGUAGE = PASCAL,
CHARTBL = PASCAL CHARTBL,
IDENT OPTBL = PASCAL IDENT OPTBL,
OPCHAR_OPTBL = PASCAL OPCHAR_OPTBL,
NUMBER_TABLE = PASCAL NUMBER_TABLE,
PRIMARY_TABLE = PASCAL PRIMARY_TABLE,
SUBSCR_TERMS = PASCAL_SUBSCR_TERM_TBL,
PRIDTBC = PASCAL_PRID_TABLE,
BIF_TABLE = PASCAL_FUNCTION_TABLE,
MULTIPLE_SUBSCR = TRUE);
                                           0000000
```

00

DB

VO

Page

(14)

OPERATOR_ENTRY('=',

EQUAL .

```
D 2
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
                                                                                                                                        VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1
                                                                                                                                                                                                Page
                                                                                                                                                                                                      (14)
V04-000
                                                 OPERATOR_ENTRY('^=',
OPERATOR_ENTRY('<=',
OPERATOR_ENTRY('>=',
OPERATOR_ENTRY('&',
OPERATOR_ENTRY('&',
                                                                                                               INFIX.
INFIX.
INFIX.
INFIX.
INFIX.
                                                                                      NOT_EQUAL,
LSS_EQUAL,
GTR_EQUAL,
BIT_AND,
   BIT OR,
                         3056
3057
3058
3059
                                        Define the Lexical Token Entry for the PL/I dereference operator "->".
                                        This token is lexically scanned separately.
                                     BIND
                         3060
3061
3062
3063
3064
3065
                                           PLI_ARROW_TOKEN =
                                                 OPERATOR_ENTRY('->',
                                                                                       PLI_DEREF,
                                                                                                               INFIX, 0, 0, PRIMARY);
                                        Define the PL/I Terminator Lexical Token Table for subscript expressions.
                                    TERMINATOR_TABLE(PLI_SUBSCR_TERM_TBL,
TERMINATOR_ENTRY(')', TERM_CLOSE),
TERMINATOR_ENTRY(':', TERM_COLON, MUST_BE_SINGLE),
TERMINATOR_ENTRY(',', TERM_COMMA));
                         3066
3067
3068
                         3069
                         3070
                         3071
3072
3073
3074
3075
3076
3077
3078
                                     ! Define the PL/I Predefined Identifier Table.
                                     PRID_TABLE(PLI_PRID_TABLE);
                                       Define the PLI Built-in function Table.
                                     BUILT_IN_FUNCTION_TABLE(PLI_FUNCTION_TABLE);
                         3080
                         3081
                         3082
3083
                                        Define the PLI Number Scanner State Table. This table defines the states
                                        of a Finite-State Machine which picks up all valid numeric constants in the
                         3084
                                        language.
                         3085
                         3086
3087
                                        Define the language PLI Number Scanner State Table. This is a finite-state
                                        machine in which each transition is of the form:
                         3088
                         3089
                                                 NUMBER_TRANSITION(character-class, action-index, next-state)
                         3090
3091
3092
3093
3094
3095
3096
3097
3098
                                        where the character-class and action-index names are automatically prefixed by "NUMST$K_CLASS_" or "NUMST$K_ACT_" by the NUMBER_TRANSITION macro.
                                     NUMBER_STATE_TABLE (PLI_NUMBER_TABLE,
                                           NUMBER_STATE(START_STATE,
NUMBER_TRANSITION(DIGIT, GO_PAST_PACK, ACCUM_INT),
NUMBER_TRANSITION(DOT, DO_NOTHING, LEADING_DOT),
NUMBER_TRANSITION(OTHER, NOT_NUMBER, END_STATE)),
                         3100
                        3101
3102
3103
3104
3105
                                           NUMBER STATE (LEADING DOT, NUMBER TRANSITION (DIGIT, GO PAST PACK FRAC, ACCUM_FRAC),
```

NUMBER_TRANSITION(OTHER, NOT_NUMBER, END_STATE)),

NUMBER_STATE (ACCUM_INT,

VO

```
E 2
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DEGPARSER
V04-000
                                                                                                                                                                                               VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                                                                                                                                                                                                                                                              Page 58 (14)
                                                                    NUMBER_TRANSITION(DIGIT, GO PAST PACK, ACCUM_INT),
NUMBER_TRANSITION(DOT, DO NOTHING, ACCUM_FRAC),
NUMBER_TRANSITION(HEXDIGIT, DO NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(B, DO NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(D, DO NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(E, DO NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(E, DO NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(OTHER, GOT_PACK_NUMBER, END_STATE)),
    NUMBER_TRANSITION(DIGIT, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(HEXDIGIT, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(B, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(D, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(E, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(E, DO_NOTHING, ACCUM_HEX),
NUMBER_TRANSITION(OTHER, GOT_NUMBER, END_STATE)),
                                                           NUMBER_STATE(ACCUM_FRAC,
NUMBER_TRANSITION(DIGIT, GO_PAST_PACK_FRAC, ACCUM_FRAC),
NUMBER_TRANSITION(DOT, BACKOP_PTRS, END_STATE),
NUMBER_TRANSITION(E, MARK_E_EXP, GET_EXPONENT),
NUMBER_TRANSITION(D, MARK_D_EXP, GET_EXPONENT),
NUMBER_TRANSITION(G, MARK_G_EXP, GET_EXPONENT),
NUMBER_TRANSITION(Q, MARK_G_EXP, GET_EXPONENT),
NUMBER_TRANSITION(OTHER, GOT_PACK_NUMBER, END_STATE)),
                                                            NUMBER_STATE(GET_EXPONENT,
NUMBER_TRANSITION(DIGIT, DO NOTHING, ACCUM_EXP),
NUMBER_TRANSITION(PLUS, DO NOTHING, GET_EXP_SIGN),
NUMBER_TRANSITION(MINUS, DO NOTHING, GET_EXP_SIGN),
NUMBER_TRANSITION(OTHER, BACKUP_PTRS, END_STATE)),
                                                            NUMBER_STATE(GET_EXP_SIGN,
NUMBER_TRANSITION(DIGIT, DO_NOTHING, ACCUM_EXP),
NUMBER_TRANSITION(OTHER, GOT_NUMBER, END_STATE)),
    NUMBER_STATE(ACCUM_EXP,
NUMBER_TRANSITION(DIGIT, DO_NOTHING, ACCUM_EXP)
                              P
                                                                      NUMBER_TRANSITION(OTHER, GOT_NUMBER, END_STATE)),
                                                            NUMBER_STATE (END_STATE
                                                                     NUMBER_TRANSITION(OTHER, GIVE_ERROR, END_STATE)));
                                                        Define the Primary Parser State Table for language PL/I. Each transition
                                                        Entry in the state table has this format:
                                                                     PRIMARY_TRANSITION(operator-code, action, next-state)
                                                        where the first parameter is the operator code which causes the transition
                                                        to be taken, the second parameter is the action routine CASE index for the
                                                         transition, and the third parameter is the next state in the finite-State
                                                        Machine.
                              9999
                                                    PRIMARY_STATE_TABLE (PLI_PRIMARY_TABLE,
                                                             PRIMARY_STATE(START_STATE
                                                                     PRIMARY_TRANSITION(GLOBAL_SLASH, START_GBL, GET_GLOBAL),
```

5E

```
DBGPARSER
V04-000
                                                                                                                                                                                                                                             16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                                                                                                                                                                                     VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1
                                                                                                                     PRIMARY_TRANSITION(BACKSLASH, START_SLASH, GOT_BACKSLASH),
PRIMARY_TRANSITION(INVOCNUM, SLASH_INVOCNUM, GOT_BACKSLASH),
PRIMARY_TRANSITION(DOT, START_DOT_PLI, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, START_SUBSCR_PLI, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PLI DEREF, START_DEREF_PLI, START_STATE),
PRIMARY_TRANSITION(PRIMARY_TERM, START_TERM, END_STATE)),
                                                         PRIMARY_STATE(GET_GLOBAL, PRIMARY_TRANSITION(PRIMARY_TERM, GBL_TERM, END_STATE)),
                                                                                                    PRIMARY_TRANSITION(BACKSLASH,
PRIMARY_TRANSITION(BACKSLASH, SLASH_SLASH, GOT_BACKSLASH),
PRIMARY_TRANSITION(INVOCNUM, SLASH_INVOCNUM, GŌT_BACKSLASH),
PRIMARY_TRANSITION(DOT, SLASH_DOT_PLI, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, SLASH_SUBSCR_PLI, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PLI DEREF, SLASH_DEREF PLI, START_STATE),
PRIMARY_TRANSITION(PRIMARY_TERM, SLASH_TERM, END_STATE)),
                                                                                                     PRIMARY_STATE(GOT_DOT,
PRIMARY_TRANSITION(DOT, DOT_DOT_PLI, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, DOT_SUBSCR_PLI, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PLI DEREF, DOT_DEREF_PLI, START_STATE),
PRIMARY_TRANSITION(PRIMARY_TERM, DOT_TERM_PLI, END_STATE)),
                                                                                                     PRIMARY_STATE(GOT_SUBSCRIPT,
PRIMARY_TRANSITION(DOT, SUBSCR_DOT_PLI, GOT_DOT),
PRIMARY_TRANSITION(SUBSCRIPT, SUBSCR_SUBSCR_PLI, GOT_SUBSCRIPT),
PRIMARY_TRANSITION(PLI_DEREF, SUBSCR_DEREF_PLI, START_STATE),
PRIMARY_TRANSITION(PRIMARY_TERM, SUBSCR_TERM_PLI, END_STATE)),
      3069
3069
3070
3071
3072
3073
3074
3076
3077
3078
3079
                                                                                                       PRIMARY_STATE(END_STATE));
                                                                                              Define the table of pointers to the parse tables for PL/I.
                                                                                    LANGUAGE_TABLES(LANGUAGE = PLI,

CHARTBL = PLI_CHARTBL,

IDENT_OPTBL = PLI_IDENT_OPTBL,

OPCHAR_OPTBL = PLI_OPCHAR_OPTBL,

NUMBER_TABLE = PLI_NUMBER_TABLE,

PRIMARY_TABLE = PLI_PRIMARY_TABLE,

SUBSCR_TERMS = PLI_SUBSCR_TERM_TBL,

PRIDTBE = PLI_PRID_TABLE,

BIF_TABLE = PEI_FUNCTION_TABLE,

MULTIPLE_SUBSCR = FALSE,

COMPONENTS_IN_PATHNAME = TRUE);
                                                          3198
3199
3200
3201
3202
3203
3204
3206
3207
                                                  00000000
      3080
3081
3082
3083
3084
3086
3086
3087
3088
```

DI

V

00

(14)

Page

(15)

Page

DI

PG PARSE TABLES

This section includes all the Lexical Scanner and Parser tables needed to scan and parse the RPG language.

Define the RPG Character Table. What is listed here is actually a list of exceptions to the Character Table for Language UNKNOWN.

```
CHAR_EXCEPTION_TABLE(RPG_CHARTBL,
CHAR_ENTRY('*', OTHER, NOTHING),
CHAR_ENTRY('<', OTHER, OPCHAR, OPCHAR_INFIX, ADDRESS_OP, TERMINATOR),
CHAR_ENTRY('>', OTHER, OPCHAR, OPCHAR_INFIX, TERMINATOR),
CHAR_ENTRY('=', OTHER, OPCHAR, OPCHAR_INFIX, TERMINATOR),
CHAR_ENTRY('*', OTHER, IDENT_ANYWHERE),
CHAR_ENTRY('*', OTHER, IDENT_ANYWHERE),
CHAR_ENTRY('S', OTHER, IDENT_ANYWHERE));
```

! Define the RPG Operator Table for operators whose names are identifiers.

```
OPERATOR_TABLE(RPG_IDENT_OPTBL,
OPERATOR_ENTRY('NOT', NOT,
OPERATOR_ENTRY('AND', AND,
OPERATOR_ENTRY('OR', OR,
OPERATOR_ENTRY('NOT', INFIX_NOT,
OPERATOR_ENTRY('NOT', INFIX_NOT', INFIX_NOT',
OPERATOR_ENTRY('NOT', INFIX_NOT', INFIX_NOT',
OPERATOR_ENTRY('NOT', INFIX_NOT', INFIX_NOT',
OPERATOR_ENTRY('NOT', INFIX_NOT', INFIX_NOT', INFIX_NOT',
OPERATOR_ENTRY('NOT', INFIX_NOT', INFIX_NOT'
```

BIND

000000

9999

00000000000000

```
RPG_NOT_EQL_TOKEN =

OPERATOR_ENTRY('NOT =', NOT_EQUAL, INFIX, 15, 15),

RPG_NOT_GTR_TOKEN =

OPERATOR_ENTRY('NOT >', LSS_EQUAL, INFIX, 15, 15),

RPG_NOT_LSS_TOKEN =

OPERATOR_ENTRY('NOT <', GTR_EQUAL, INFIX, 15, 15);
```

Define the RPG Operator Table for operators whose names are composed of operator characters such as "+", "-", or "*". This table includes operators which are part of DEBUG Primary Symbols (such as "\").

```
OPERATOR TABLE (RPG OPCHAR OPTBL,

OPERATOR ENTRY('\'. GLOBAL SLASH,
OPERATOR ENTRY('\'. BACKSLASH,
OPERATOR ENTRY('\'. SUBSCRIPT,
OPERATOR ENTRY('\'. PREFIX_GTR,
OPERATOR ENTRY('\'. PREFIX_LSS,
OPERATOR ENTRY('\'. PREFIX_EQL,
OPERATOR ENTRY('\'. OPENPAREN,
OPERATOR ENTRY('\'. DIVIDE,
OPERATOR ENTRY('\'. UNARY PLUS,
OPERATOR ENTRY('\'. UNARY PLUS,
OPERATOR ENTRY('\'. UNARY PLUS,
OPERATOR ENTRY('\'. UNARY MINUS,
OPERATOR ENTRY('\'. SUBTRACT,
OPERATOR ENTRY('\'. SUBTRACT,
OPERATOR ENTRY('\'. GTR_THAN,
OPERATO
```

D

Page 62 (15)

D

Page

(16)

DBGPARSER V04-000										K 2 16-Sep-19 14-Sep-19	84 02:10 84 12:17):13 :30	VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1	Page 64 (17)
3262 3263 3264 3265 3266 3267 3269 3271 3272 3273 3274 3275 3276 3278 3278 3279	333338845 333338845 333338885 3333333333	f F	INPUT OUTPU BEG RET	Thi to the IS NON	force sou								le purpose is a tables before odule.	
; 3200	3371		END								****	2000	*****	
											IDENT	1004	ARSER -000\	
											.PSECT	DBG\$	PLIT, NOWRT, SHR, PIC, 0	
									00 03	00000 P.AAA: 00002	.BYTE	31 0		;
			00 54	52	00 45	00 56	00 4E	00 4F	00 00 43 07 00 03	00004 0000C	.ASCII	₹7>\	0, 0, 0, 0, 0 CONVERT\	
			00	00	00	00 4F	00 50	00 45	00 00 43 07 00 03 0032 00 00	00014 P.AAB: 00016 00018	.WORD	56	0, 0, 0, 0, 0, 0 ĎEPÓSIÍ\	
			54	49	23	41	50	45	00 00 44 07 00 02	00028 P.AAC:	.ASCII	2.0	DEPOSITY	
		59	00 54	00 49	00 54	00 4E	00 45	00 44	00 00 49 08 00 02 00 02 00 00 4E 08 00 02	00000 P.AAA: 00002 00004 00006 00014 P.AAB: 00016 00018 00020 00028 P.AAC: 00024 00034 00035 P.AAC: 00049 00052 P.AAE: 00054 00056 00056 00056 00056 00057 P.AAF: 00069 00068 00077 00077 00079 00081 00083 P.AAH: 00087 P.AAH:	BYTE WORD BYTE BYTE	<8>1	IDENTITY'	
			00 53	00 4E	00 4F	00 43	00	00	00 00	0003F 00041	.WORD	66	, 0, 0, 0, 0, 0 NEGCONST\	
		54	>>	4E	41	45	41	45	00 00 4E 08 00 02	00052 P.AAE:	.ASCII	2,0	NEGCONSTA	
		54	00 53	00 4E	00 4F	00	00 53	00 4F	00 00 50 08 00 02	00056 0005E	BYTE	0, 0	0, 0, 0, 0, 0, 0 POSCONST\	
								00	00 02	00067 P.AAF:	.WORD	2.0		
			00	00	00	00	00	00	00 00 50 08 00 02 00 02 00 00 20 01 00 02	00073 00075 P.AAG:	ASCII	\$1>\ 2.0	<u>-</u> \0. 0. 0. 0. 0	
			00	00	00	00	00	00	0004 00 00 2B 01 41 42 01 25 05	00077	.ASCII .BYTE .WORD .BYTE		. 0. 0. 0. 0. 0	
							45	53	00 00 2B 01 41 42	00081 00083 P.AAH:	ASCII BYTE ASCII	<1>\	E\	
					45	4E	49	40	25 05	00087 P.AAJ:	.ASCII	1	ZL INE \	

DCDAPSE D								L 2	0200 4
DBGPARSER V04-000								16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 6 (17
			4E	49	40	25	01	0008E P.AAK: .BYTE 1 0008F .ASCII <4>\%LIN\	1
				49	40	25	01	00094 P.AAL: .BYTE 1 00095 .ASCII <3>\%LI\	
	40	45	42	41	40	25	90	00099 P.AAM: .BYTE 2 0009A .ASCII <6>\%LABEL\	
		45	42	41	40	25	02	00099 P.AAM: .BYTE 2 0009A .ASCII <6>\%LABEL\ 000A1 P.AAN: .BYTE 2 000A2 .ASCII <5>\%LABE\	
			42	41	40	25	32625242335838383838383838383838383838383838383	000A8 P.AAU: .BYTE 2 000A9 .ASCII <4>\%LAB\	
				41	40	25	02	000AE P.AAP: .BYTE 2 000AF .ASCII <3>\%LA\	1
		45	40	41	4E	25	03	000B3 P.AAQ: .BYTE 3 000B4 .ASCII <5>\%NAME\	
				30	52	25	08	000BA P.AAR: .BYTE 8 000BB .ASCII <3>\%RO\ 000BF P.AAS: .BYTE 8 000CO .ASCII <3>\%R1\	
				31	52	25	08	000BF P.AAS: .BYTE 8 000C0 .ASCII <3>\%R1\	
				32	52	25	08	000C4 P.AAT: .BYTE 8 000C5 .ASCII <3>\%R2\	1
				33	52	25	08	000C9 P.AAU: .BYTE 8 000CA .ASCII <3>\%R3\	:
				34	52	25	08	000CE P.AAV: .BYTE 8 000CF .ASCII <3>\%R4\	
				35	52	25	03	00003 P.AAW: .BYTE 8 00004 .ASCII <3>\%R5\	
				36	52	25	03	00008 P.AAX: .BYTE 8 00009 .ASCII <3>\%R6\	
				37	52	25	03	000DD P.AAY: .BYTE 8 000DE .ASCII <3>\%R7\ 000E2 P.AAZ: .BYTE 8 000E3 .ASCII <3>\%R8\	
				38	52	25		000E2 P.AAZ: .BYTE 8 000E3 .ASCII <3>\%R8\	
				39	52	25	03	000E7 P.ABA: .BYTE 8 000E8 .ASCII <3>\%R9\	
			30	31	52	25	04	000E7 P.ABA: .BYTE 8 000E8 .ASCII <3>\%R9\ 000EC P.ABB: .BYTE 8 000ED .ASCII <4>\%R10\ 000F2 P.ABC: .BYTE 8 000F3 .ASCII <4>\%R11\ 000F8 P.ABD: .BYTE 8 000F9 .ASCII <4>\%R12\ 000FF P.ABE: .BYTE 8 .ASCII <4>\%R13\ 000FF .ASCII <4>\%R13\ 00104 P.ABF: .BYTE 8	
			31	31	52	25	04	000F2 P.ABC: .BYTE 8 000F3 .ASCII <4>\%R11\	
			32	31	52	25	04	000F8 P.ABD: .BYTE 8 000F9 .ASCII <4>\%R12\	
			33	31	52	25	04	000FE P.ABE: .BYTE 8 000FF .ASCII <4>\%R13\	
			34	31	52	25	04	00104 P.ABF: .BYTE 8 00105 .ASCII <4>\%R14\	
			35	31	52	25	08	0010A P.ABG: .BYTE 8 0010B .ASCII <4>\%R15\\ 00110 P.ABH: .BYTE 8 00111 .ASCII <3>\%AP\\ 00115 P.ABI: .BYTE 8 00116 .ASCII <3>\%FP\\ 00116 .ASCII <3>\%FP\	
				50	41	25	03	00110 P.ABH: .BYTE 8 00111 .ASCII <3>\%AP\	
				50	46	25	03	00115 P.ABI: .BYTE 8 00116 .ASCII <3>\%FP\	
				50	53	25	08	0011A P.ABJ: .BYTE 8 0011B .ASCII <3>\%SP\	
				43	50	25	8384848484848483838383848	0011A P.ABJ: .BYTE 8 0011B .ASCII <3>\%SP\ 0011F P.ABK: .BYTE 8 00120 .ASCII <3>\%PC\ 00124 P.ABL: .BYTE 8 00125 .ASCII <4>\%PSL\ 0012A P.ABM: .BYTE 8	
			40	53	50	25	08	00124 P.ABL: .BYTE 8 00125 .ASCII <4>\%PSL\	

43 4F 4C 54

				M 2 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 66 (17)
	30	72	25		·
	31	72	25	08 0012F P.ABN: .BYTE 8 03 00130 .ASCII <3>\%r1\	
	32	72	25	08 00134 P.ABO: .BYTE 8 03 00135 .ASCII <3>\%r2\	
	33	72	25	03 0012B	
	34	72	25	08 0013E P.ABQ: .BYTE 8 03 0013F .ASCII <3>\%r4\	
	35	72	25	03 0012B	
				08 00148 P.ABS: .BYTE 8	
	36	72	25	08 0014D P.ART: .RYTE 8	
	37	72	25	08 0014D P.ABT: .BYTE 8 03 0014E .ASCII <3>\%r7\ 08 00152 P.ABU: .BYTE 8 03 00153 .ASCII <3>\%r8\ 08 00157 P.ABV: .BYTE 8	
	38	72	25	08 00152 P.ABU: .BYTE 8 03 00153 .ASCII <3>\%r8\ 08 00157 P.ABV: .BYTE 8	
	39	72	25	05 00138 .ASCII <3>\%r9\	
30	31	72	25	04 0015D .ASCII <4>\%r10\	
31	31	72	25	08 00162 P.ABX: .BYTE 8 04 00163 .ASCII <4>\%r11\	
32	31	72	25	08 00168 P.ABY: .BYTE 8 04 00169 .ASCII <4>\%r12\	
33	31	72	25	08 0016E P.ABZ: .BYTE 8 04 0016F .ASCII <4>\%r13\	:
34	31	72	25	08 00174 P.ACA: .BYTE 8 04 00175 .ASCII <4>\%r14\	
35	31	72	25	08 0017A P.ACB: .BYTE 8 04 0017B .ASCII <4>\%r15\	
	70	61	25	08 00180 P.ACC: .BYTE 8 03 00181 .ASCII <3>\%ap\ 08 00185 P.ACD: .BYTE 8	
	70	66	25	08 00185 P.ACD: BYTE 8 03 00186 .ASCII <3>\%fp\	
				03 00186 .ASCII <3>\%fp\ 08 0018A P.ACE: .BYTE 8	
	70	73	25	08 0018A P.ACE: .BYTE 8 03 0018B .ASCII <3>\%sp\ 08 0018F P.ACF: .BYTE 8 03 00190 .ASCII <3>\%pc\ 08 00194 P.ACG: .BYTE 8	
	63	70	25	03 00190 .ASCII <3>\%pc\ 08 00194 P.ACG: .BYTE 8	
60	73	70	25	04 00195 .ASCII <4>\%psl\	
43	45	44	25	04 0019B .ASCII <4>\%DEC\	
58	45	48	25	04 001A1 .ASCII <4>\%HEX\	
54	43	4F	25	04 001A7 .ASCII <4>\%0CT\	
4E	49	42	25	07 001AC P.ACK: .BYTE 7 04 001AD .ASCII <4>\%BIN\ 08 001B2 P.ACL: .BYTE 8	
52	55	43	25	08 001B2 P.ACL: .BYTE 8 07 001B3 .ASCII <7>\%CURLOC\	
52	55	43	25	08 001BB P.ACM: .BYTE 8 07 001BC .ASCII <7>\%CURVAL\	
45	52	50	25	08 001C4 P.ACN: .BYTE 8 08 001C5 .ASCII <8>\%PREVLOC\	
58	45	4E	25	08 001CE P.ACO: .BYTE 8 08 001CF .ASCII <8>\%NEXTLOC\	
			•		

DBGPARSE V04-000	:R		N 2 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 Pag 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	e 67
		54 4E 43 52 41 50 25	08 001D8 P.ACP: .BYTE 8 07 001D9 .ASCII <7>\%PARCNT\ 001E1 .BLKB 3 .LONG 59 .OO4 001E8 P.AAI: .LONG 4, 11, 17, 22, 30, 37, 43, 48, 55, 60, -	
00000025 00000046 00000087 000000A7 000000C5 000000E5 00000107 00000129	0000001E 0000005F 00000081 00000000 00000000 00000000 00000102 00000123 00000155	00000016 00000011 0000000B 0000 0000003C 00000037 00000030 0000 0000005A 00000055 00000050 0000 0000007B 00000075 0000006F 0000 0000009C 00000097 00000092 0000 000000BB 000000B6 000000B1 0000 000000PD 000000PT 000000FT 0000 000000FD 000000FT 000000F1 0000 0000011D 00000117 00000111 0000 0000014B 00000141 00000138 0000	104B 00218	
45 73	70 79 45	00	001 002D6 .WORD 1	
65 72	70 78 65	6E 6F 69 73	73 002EF 04 002F4 P.ACR: BYTE 4.2	
73 73	65 72 70	78 65 20 66 6F 20 64 6E 65 6F 01	002 002F6 .WORD 2 C8 002F8 .BYTE -56, 2, 0, 0, 0, 0, 0, 0 11 00300 .ASCII <17>\end of expression\ 69 0030F 04 00312 P.ACS: .BYTE 4, 1	
		01	04 00312 P.ACS: .BYTE 4, 1	
60	6F 62 6D	79 73 20 66 6F 20 64 6E 65	00 00316	
		00 00 00 00 00 00 08 43 45 44 25	001 00314	
		00 00 00 00 00 00 08 58 45 48 25	001 00314	
		00 00 00 00 00 00 08 54 43 4F 25	02 0034E P.ACV: BYTE 2, 2 036 00350	
		00 00 00 00 00 00 00 08 54 43 4F 25 02 00 00 00 00 00 00 00 08	BE 00352 .BYTE -66, -56, 0, 0, 0, 0, 0, 0 ASCII <4>\%0CT\ 02 0035F P.ACW: .BYTE 2, 2 037 00361 WORD 55 BE 00363 BYTE -66, -56, 0, 0, 0, 0, 0 0	
		00 00 00 00 00 00 C8	037 00361 .WORD 55 BE 00363 .BYTE -66, -56, 0, 0, 0, 0, 0, 0 04 0036B .ASCII <4>\%BIN\ 01 00370 P.ACX: .BYTE 1, 0	
		43 4F 4C 52 55 43 25	001 00372 .WORD 1 00 00374 .BYTE 0, 0, 0, 0 07 00378 .ASCII <7>\%CURLOC\ 01 00380 P.ACY: BYTE 1, 0	
	•	4C 41 56 52 55 43 25	BE 00341 04 00349 02 0034E P.ACV: BYTE 2, 2 036 00350 BE 00352 BYTE -66, -56, 0, 0, 0, 0, 0, 0 BE 00352 ASCII <4>\%ACTI 04 \%ACTI 05 00354 BYTE -66, -56, 0, 0, 0, 0, 0 ASCII <4>\%ACTI 06 00355 BYTE -66, -56, 0, 0, 0, 0, 0 ASCII <4>\%ACTI 07 00361 BYTE -66, -56, 0, 0, 0, 0, 0 ASCII <4>\%BIN\ 01 00370 BYTE 1, 0 BYTE 0, 0, 0, 0 ASCII <7>\%CURLOC\ 07 00378 BYTE 1, 0 BYTE 0, 0, 0, 0 ASCII <7>\%CURLOC\ 07 00388 ASCII <7>\%CURVAL\ 07 00388 BYTE 1, 0	
		43 4F 4C 56 45 52 50 25	001 00392 .WORD 1 00 00394 .BYTE 0, 0, 0 0 08 00398 .ASCII <8>\%PRÉVLOC\ 02 003A1 P.ADB: .BYTE 2, 0	

DBGPARSER V04-000	B 3 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 Page 68 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1 (17)	В
00 00 00	00 00 00 00 08 28 003A5	
00 00 00	00 00 00 08 28 003B3	
00 00 00	0006 003BF .WORD 6 000 00 00 0A 0A 003C1 .BYTE 10, 10, 0, 0, 0, 0, 0 0 2B 01 003C9 .ASCII <1>\+\ 00 03 003CB P.ADE: .BYTE 3, 0 0007 003CD .WORD 7	
00 00 00	0007 003CD .WORD 7 000 00 00 0A 0A 003CF .BYTE 10, 10, 0, 0, 0, 0, 0, 0 2D 01 003D7 .ASCII <1>\-\	
00 00 00	0004 003DB .WORD 4 000 00 00 C8 14 003DD .BYTE 20, -56, 0, 0, 0, 0, 0 2B 01 003E5 .ASCII <1>\+\ 00 02 003E7 P.ADG: .BYTE 2, 0 0005 003E9 .WORD 5	
. 00 00 00	00 00 00 C8 14 003EB .BYTE 20, -56, 0, 0, 0, 0, 0, 0 2D 01 003F3 .ASCII <1>\-\ 00 03 003F5 P.ADH: .BYTE 3, 0 0008 003F7 .WORD 8	
00 00 00	0008 003F7 .WORD 8 000 00 00 1E 1E 003F9 .BYTE 30, 30, 0, 0, 0, 0, 0, 0 2A 01 00401 .ASCII <1>*\ 00 03 00403 P.ADI: .BYTE 3, 0 0009 00405 .WORD 9	
00 00 00	0009 00405 .WORD 9 000 00 00 1E 1E 00407 .BYTE 30, 30, 0, 0, 0, 0, 0, 0 2F 01 0040F .ASCII <1>\/\ 02 04 00411 P.ADJ: .BYTE 4, 2 0031 00413 .WORD 49	
00 00 00	00 00 00 32 C8 00415	
00 00 00	000 00 00 32 C8 00415	
00000364 00000356 00000348 0000033A 000003AA 0000039C 0000038E	3C 01 00415 02 02 0041F P.ADK: BYTE 2, 2 000B 00421	
000003AA 0000039C 0000038E	00000000 0046C .LONG C ; 00470 P.ADM: .BLKB 0 00 00 00470 P.ADO: BYTE 0 0	
	00 00 00 00 00474 .BYTE 0, 0, 0, 0	
	00000001 0047C .LONG 1 000003ED 00480 P.ADN: .LONG 1005 01 00 00484 P.ADQ: .BYTE 0, 1	

Page 69 (17)	
(17)	

DE

			C 3 16-Sep-19 14-Sep-19	84 02:10:13 VAX-11 Bliss-32 V4.0-7 84 12:17:30 [DEBUG.SRC]DBGPARSER.B	742 332;1
	00	00 00 00 30 00	00488	.WORD 4 .BYTE 0, 0, 0, 0 .ASCII <1>\=\	
		00000001 00000401 01 00	00494 P.ADP: 00498 P.ADS: 0049A	.WORD 5 .BYTE 0, 0, 0, 0 .ASCII <1>\=\ .BLKB 2 .LONG 1 .LONG 1025 .BYTE 0, 1 .WORD 5 .BYTE 0, 0, 0, 0 .ASCII <2>\DON .BLKB 1	
	00	00 00 00 4F 44 02	0049C 004A0 004A3	.BYTE 0, 0, 0, 0 .ASCII <2>\DO\ .BLKB 1	
		00000001 00000415 01 00	004A8 P.ADR: 004AC P.ADU: 004AE	LONG 1 LONG 1045 BYTE 0, 1 .WORD 6	
	4E 45	00000000	00480 00484 00489	BYTE 0, 0, 0, 0 ASCII <4>\THEN\ BLKB 3	
		00000429 00 00	004C0 P.ADT: 004C4 P.ADW: 004C6	LONG 1065 BYTE 0, 0 WORD 1	
	00	00 00 00 2C 00 02 00	004CC 004CE P.ADX:	LONG 1045 .BYTE 0, 1 .WORD 6 .BYTE 0, 0, 0, 0 .ASCII <4>\THEN\ .BLKB 3 .LONG 1 .LONG 1065 .BYTE 0, 0, 0, 0 .WORD 1 .BYTE 0, 0, 0, 0 .ASCII <1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	
	00	00 00 00 3A 01	00402	.BYTE 0, 0, 0, 0 .ASCII <1>\:\ .LONG 2	
	000004	00 00	004E4 P.ADZ:	LONG 1089, 1099 BYTE 0, 0 WORD 1	
	00	00 00 00 2C 01 01 00	004EC P.AEA:	.BYTE 0, 0, 0, 0 .ASCII <1>\ .BYTE 0, 1 .WORD 7	
	4E 45	00 00 00 48 57 04 01 00	004F2 004F6 004FB P.AEB: 004FD 004FF	.BYTE 0, 0, 0, 0 .ASCII <4>\WHEN\ .BYTE 0, 1 .WORD 5	
	00	4F 44 02	00503	.BYTE 0, 0, 0, 0 .ASCII <2>\DO\ .BLKB 2	
00000478	000004	6B 00000461 00 00 00 00	0050C P.ADY:	LONG 1121, 1131, 1144	
	00	00 00 00	00510 00520 00522	.BLKB 2	
		0000000 00000499 00 00	00528 P.AEC: 0052C P.AEF: 0052E	.LONG 1173 .BYTE 0, 0 .WORD 1	
	00	00 00 00 2C 00 01 00	00530	.BYTE 0, 0 .WORD 1 .BYTE 0, 0, 0, 0 .ASCII <1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	

DBGPARSER V04-000

		16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 70 (17)
00 0 000004B3	0002 0 00 00 00 0 29 01 0 00000002 0	. WORD 2	
	01 00 0 0000 0 00 00 00 0 0 54 02 0	Second State	
00 0	00000001 0 000004C9 0 01 00 0 000E 0	10558 .LONG 1 1055C P.AEH: .LONG 1225 10560 P.AEK: .BYTE 0, 1 10562 .WORD 14	
5	00 00 00 00 69 42 02 0 01 00 0 0005 0 00 00 00 0 0F 44 02 0	00564	
000004E8	00000002 000004DD 00 00	00573 .ASCII <2>\DO\ 00576 .BLKB 2 .DO578 .LONG 2 .DO57C P.AEJ: .LONG 1245, 1256 .DO584 P.AEN: .BYTE 0, 0	
00 0	00 00 00 00 00 00 00 00 00 00 00 00 00	00586 .WORD 1 00588 .BYTE 0, 0, 0, 0 0058C .ASCII <1>\ 0058E P.AEO: .BYTE 0, 0 00590 .WORD 10	
00 0 0000050B	000A 000 00 00 00 00 00 00 00 00 00 00 0	00596 P.AEM: .LONG 1281, 1291	
00 0	0001 0 00 00 0 2C 01 0 01 00 0	05A8 .BYTE 0, 0, 0, 0	
	0002 0 00 00 0 5D 01 0 00 00 0	005B2	
	00000003	OSAE P.AER: .BYTE 0, 1 OSBO	
00000535 0000052B 000003FD 0000043D 00000425	00000411 0	00504 TERM_POINTER_TBL: .LONG 1005, 1021 .DOSDC .BYTE 0[8] .DOSE4 .LONG 1041, 1061, 1085 .DOSFO .BYTE 0[4]	
00000489 000004F9 000004D9 000004C1	00000459 0 000004A5 0	LONG 1005, 1021 DOSDC BYTE 0[8] DOSE4 LONG 1041, 1061, 1085 DOSFO BYTE 0[4] DOSF4 LONG 1113, 1161 DOSFC BYTE 0[4] DOSFC BYTE 0[4]	

DI

DBGPARSER V04-000			F 3 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 Page 7 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32:1 (17
	00 00 00	00 00 00 14 14 52 4F 02 00 03	OOAB8 P.AFD: BYTE 3, 0 OOABC .WORD 25 OOACC .BYTE 20, 20, 0, 0, 0, 0, 0 OOACC .ASCII <2>\OR\ OOACC .WORD 26 OOACC .WORD 26 OOACC .BYTE 10, 10, 0, 0, 0, 0, 0 OOACC .BYTE 10, 10, 0, 0, 0, 0, 0 OOACC .BYTE 10, 10, 0, 0, 0, 0, 0 OOACC .BYTE 3, 0 .WORD 26 OOACC .BYTE 3, 0 .WORD 27 OOACC .WORD 27 OOACC .BYTE 10, 10, 0, 0, 0, 0, 0 .WORD 27 OOACC .BYTE 10, 10, 0, 0, 0, 0, 0 .ASCII <3>\EQV\ OOACC .BYTE 10, 10, 0, 0, 0, 0, 0 .ASCII <3>\EQV\ OOACC .BUKB 1 .LONG 11 OOACC P.AEU: .LONG 2485, 2501, 2517, 2533, 2549, 2565, 2581, -;
	00 00 00	00 00 00 0A 0A 52 4F 58 03 00 03 00 01B 00 00 00 0A 0A 56 51 45 03	00AC9 .WORD 26 00ACB .BYTE 10, 10, 0, 0, 0, 0, 0 00AD3 .ASCII <3>\XOR\ 00AD7 P.AFF: .BYTE 3, 0
	00 00 00	00 00 00 0A 0A 56 51 45 03	UNID
00000A05 000009F5 00000A54	000009E5 000009D5 00000A44 00000A35	000009C5 000000B5 00000A25 00000A15 01 02	00804 2597, 2613, 2628, 2644 :
	00 00 00	00 00 00 00 00 5C 01 01 03	00B18 P.AFH: BYTE 2, 1 00B1A
	00 00 00	00 00 00 00 00 5C 01 01 03	00B28 .WORD 3 00B2A .BYTE 0, 0, 0, 0, 0, 0, 0 00B32 .ASCII <1><92> 00B34 P.AFJ: .BYTE 3, 1
	00 00 00	00 00 00 00 00 2E 01 01 04	00B36 .WORD 5 00B38 .BYTE 0, 0, 0, 0, 0, 0, 0 00B40 .ASCII <1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
	00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00	00B44 .WORD 4 00B46 .BYTE 0, 0, 0, 0, 0, 0, 0 00B4E .ASCII <1>(\ 00B50 P.AFL: .BYTE 4, 1
	00 00 00	00 00 00 00 00 5B 01 01 04	00B52 .WORD 4' 00B54 .BYTE 0, 0, 0, 0, 0, 0, 0 00B5C .ASCII <1>(1)
	00 00 00	00 00 00 00 00 5E 01 00 03	00B60 .WORD 7 00B62 .BYTE 0, 0, 0, 0, 0, 0
	00 00 00	00 00 00 3C 3C 3C 2F 2F 02 00 03	00B6C P.AFN: .BYTE 3, 0 00B6E .WORD 35 00B70 .BYTE 60, 60, 0, 0, 0, 0, 0
	00 00 00	00 00 00 3C 3C 3C 26 01 00 03 0006	00B70
	90 00 00	00 00 00 3C 3C 2B 01 00 03	OOB6C P.AFN: BYTE 3, 0 OOB6E
	00 00 00	00 00 00 3C 3C 2D 01 00 02	00B95

D

							G 3 16-Sep-19 14-Sep-19	84 02:10:13 VAX-11 Bliss-32 V4.0-742 84 12:17:30 [DEBUG.SRCJDBGPARSER.B32;1	Page 73
00	00	00	00	00	00	0004 C8 46 2B 01 00 02	00BA7 00BA9 00BB1 00BB3 P.AFS:	.WORD 4 .BYTE 70, -56, 0, 0, 0, 0, 0 .ASCII <1>\+\ .BYTE 2, 0	
00	00	00	00	00	00	0005 C8 46 20 01 00 03	00BB5 00BB7 00BBF	.WORD 5 .BYTE 70, -56, 0, 0, 0, 0, 0	
00	00	00	00	00	00	0008 50 50 2A 01	00BC1 P.AFT: 00BC3 00BC5 00BCD	ASCII <1>*\	
00	00	00	00	00	00	00 03 0009 50 50 2F 01 00 03	00BCF P.AFU: 00BD1 00BD3 00BDB	.BYTE 3, 0 .WORD 9 .BYTE 80, 80, 0, 0, 0, 0, 0	
00	00	00	00	00	00 2A	000A 5C 5A 2A 02 00 03	OOBDD P.AFV: OOBDF OOBE1 OOBE9	.BYTE 3, 0 .WORD 10 .BYTE 90, 92, 0, 0, 0, 0, 0 .ASCII <2>**\ .BYTE 3, 0 .WORD 13	
00	00	00	00	00	00	32 32 30 01	OOBEC P.AFW: OOBEE OOBFO OOBF8	BYTE 3, 0 .WORD 13 .BYTE 50, 50, 0, 0, 0, 0, 0	
00	00	00	00	00	00 3E	00 03 000E 32 32 30 02 00 03	00BFA P.AFX: 00BFC 00BFE 00C06 00C09 P.AFY:	.BYTE 50, 50, 0, 0, 0, 0, 0 .ASCII <1>\=\ .BYTE 3, 0 .WORD 14 .BYTE 50, 50, 0, 0, 0, 0, 0 .ASCII <2>\<>\	
00	00	00	00	00	00 30	000E 32 32 2F 02 00 03	00C0B 00C0D 00C15	.BYTE 3, 0 .WORD 14 .BYTE 50, 50, 0, 0, 0, 0, 0 .ASCII <2>\/=\ .BYTE 3, 0	
00	00	00	00	00	00	000F 32 32 3E 01 00 03	00C1A	.WORD 15 .BYTE 50, 50, 0, 0, 0, 0, 0	
00	00	00	00	00	00 30	00 03 32 32 3E 02 00 03	00C24 00C26 P.AGA: 00C28 00C2A 00C32 00C35 P.AGB:	.BYTE 3, 0 .WORD 17 .BYTE 50, 50, 0, 0, 0, 0, 0 .ASCII <2>\>=\ .BYTE 3, 0 .WORD 19	
00	00	00	00	00	00	32 32 3C 01	00C37 00C39 00C41	ASCII <1>\<\	
00	00	00	00	00	00 30	00 03 0015 32 32 30 02 02 02	00C43 P.AGC: 00C45 00C47 00C4F 00C52 P.AGD:	.BYTE 3, 0 .WORD 21 .BYTE 50, 50, 0, 0, 0, 0, 0	
00	00	00	00	00	00	000B C8 05 28 01 02 04	00C56 00C5E	.BYTE 2, 2 .WORD 11 .BYTE 5, -56, 0, 0, 0, 0, 0 .ASCII <1>\(\)	
00	00	00	00	00	00	000C 06 C8 29 01	00C60 P.AGE: 00C62 00C64 00C6C	.WORD 12 .BYTE -56, 6, 0, 0, 0, 0, 0	
					0	0000018	00C6E 00C70	.BLKB 2 .LONG 24	:

DIV

DBGPARSER V04-000					H 3 16-Sep-1984 02 14-Sep-1984 12	10:13	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page 7:
00000B30 00000B86 00000BDD	0000ACD 0000B22 0000BCF	00000Bf 00000B69 00000BC0	0000085A 00000B5A 00000B82	00000AA3 00000A95 00000B4C 00000B3E 00000BA3 00000B95 01 00 00 00 00 00 00 00 00 00 00 00 00 00	OOC74 P.AFG: .LONG OOC8C OOCA4 OOCBC OOCD6 .WORI OOCD6 .ASC: OOCDE P.AGH: .BYTI OOCE0 .ASC: OOCE2 .BYTI OOCE2 .BYTI OOCE4 .ASC: OOCE5 P.AGI: .BYTI OOCE6 .ASC: OOCE6 .ASC: OOCE6 .ASC: OOCE6 .ASC: OOCE7 .BYTI	2709 2808 2906 3008 0, 1	2723. 2737. 2751. 2765. 2779. 2 2822. 2836. 2850. 2864. 2878. 2 2921. 2935. 2950. 2965. 2979. 2 3023. 3037. 0. 0 0. 0 0. 0 3163. 3173. 3183	793

DV

DBGPARSER V04-000		J 16-Sep-19 14-Sep-19	84 02:10:13 84 12:17:30	VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1	Page 76
	0021 001 001 0021 0000 0021 0000000 0020 0000000 001 001	0008A 0008C 0008D 00090 00091 00092 00096 00096 00096 00096 00090 000000	WORD 33 BYTE 31 BYTE 33 BYTE 3		

DBGPARSER V04-000	K 3 16-Sep-1984 02:10:13 14-Sep-1984 12:17:30	VAX-11 Bliss-32 V4.0-742 Page 77 EDEBUG.SRCJDBGPARSER.B32;1 (17)
V04-000	00000000 00DDE	ĬĎĒBŪĠ. SŘĊĴĎBĠĎAŘŠĖŘ. BŠŽ:1
00000C7D 00000D1D 00000C8D 00000BF1 00000	0020 00E1A .WORD 32 00000000 00E1C .LONG 0 00000000 00E20 .LONG 0	3. 2665. 3057. 3213. 3357. 3197. 3353
00000000 00000000 00000001 00000000 00000	00000000 00E54 00000009 00E58 .LONG 9	3, 2665, 3057, 3213, 3357, 3197, 3353, - 7, 0, 1, 0, 0, 0
3C00A800 2F00A800 26000800 230000C0 5F000 2E058830 3E042	1029	311424, 1593835698, 587202752, - 536256, 788572160, 1006675968, - 3682560, 1040459776, 772114480
00 00 00 00 0	00 02 00E80 P.AGR: .BYTE 2.0 0017 00E82 .WORD 23 00 00 C8 3C 00E84 .BYTE 60. 54 4F 4E 03 00E8C .ASCII <3>0 00 02 00E90 P.AGS: .BYTE 2.0	-56, 0, 0, 0, 0, 0
00 00 00 00 0	002B 00E92 .WORD 43	-56, 0, 0, 0, 0, 0

DBGPARSER V04-000			L 3 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 Page 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	e 78
	00 00	00 00 00 32 44 4F 4D	32 00EA4 .BYTE 50, 50, 0, 0, 0, 0, 0 ;	
	00 00 00	00 00 00 32 40 45 52	03	
	00 00 00	00 00 00 0A 44 4E 41	03 00EC0 P.AGV: .BYTE 3, 0 18 00EC2 .WORD 24 0A 00EC4 .BYTE 10, 10, 0, 0, 0, 0, 0 03 00ECC .ASCII <3>\AND\ 03 00ED0 P.AGW: .BYTE 3, 0 19 00ED2 .WORD 25	
	00 00 00	00 00 00 0A 00 00 00 00 00 00 00 00 00 0	19 00ED2 .WORD 25 0A 00ED4 .BYTE 10, 10, 0, 0, 0, 0, 0 0 02 00EDC .ASCII <2>\OR\ 03 00EDF P.AGX: .BYTE 3, 0	
	00 00 00		OA OOEES .BYTE 10, 10, 0, 0, 0, 0, 0	
00000E4D 00000E3D	00000E2D 00000E1D	00000E0D 000000 00000E 01	07 00EF0 FD 00EF4 P.AGQ: .LONG 7 5C 00F0C 02 00F10 P.AGZ: .BYTE 2, 1 02 00F12 .WORD 2	
	00 00 00	00 00 00 00 5C 01	02 00F12 .WORD 2 00 00F14 .BYTE 0, 0, 0, 0, 0, 0, 0 01 00F1C .ASCII <1><92>	
	00 00 00	00 00 00 00 50 01	03 00F20 .WORD 3 00 00F22 .BYTE 0, 0, 0, 0, 0, 0, 0 01 00F2A .ASCII <1><92> :	
	00 00 00	00 00 00 00 2E 01	00 00F30 .BYTE 0, 0, 0, 0, 0, 0, 0 ;	
	00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00	00 00F30	
	00 00 00	00 00 00 C8 28 02	0B 00F4A .WORD 11 05 00F4C .BYTE 5, -56, 0, 0, 0, 0, 0 0 01 00F54 .ASCII <1>\(\)	
	00 00 00	00 00 00 06	04 00F56 P.AHE: .BYTE 4, 2 0C 00F58 .WORD 12 C8 00F5A .BYTE -56, 6, 0, 0, 0, 0, 0 0 01 00F62 .ASCII <1>\)\ 03 00F64 P.AHF: .BYTE 3, 0	
	00 00 00	00 00 00 3C 2A 2A 00	01 00F38	
	00 00 00	00	01	

DBGPARSER V04-000			M 3 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 Page 79 14-Sep-1984 12:17:30 [DEBUG.SRCJDBGPARSER.B32;1 (17)
	00 00 00	00 00 00 32 32 2F 01 00 02	OOF 83 OOF 85 OOF 85 OOF 80 OOF 87 OOF 87 OOF 87 OOF 91 OOF 93 OOF 93 OOF 93 OOF 94 OOF 98 OOF 99 OOF 99 OOF 90 OOF 90 OOF 90 OOF 90 OOF 91 OOF 91 OOF 92 OOF 93 OOF 94 OOF 95 OOF 96 OOF 97 OOF 98 OO
	00 00 00	00 00 00 C8 28 2B 01 00 02	00F91 .WORD 4 00F93 .BYTE 40, -56, 0, 0, 0, 0, 0 00F9B .ASCII <1>\+\ 00F9D P.AHJ: .BYTE 2, 0
	00 00 00	00 00 00 C8 28 20 01 00 03	00FA1 .BYTE 40, -56, 0, 0, 0, 0, 0
	00 00 00	00 00 00 1E 1E 2B 01 00 03	OOF AD .WORD 6 OOF AF .BYTE 30, 30, 0, 0, 0, 0, 0 OOF BP P.AHL: .BYTE 3, 0 OOF BB .WORD 7 OOF BD .BYTE 30, 30, 0, 0, 0, 0, 0 OOF C5 .ASCII <1>\-\ OOF C7 P.AHM: .BYTE 3, 0
		00 00 00 1E 1E 2D 01 00 03 0023	OOFB7 OOFB9 P.AHL: .BYTE 3, 0 .WORD 7 OOFBD .BYTE 30, 30, 0, 0, 0, 0, 0 OOFC5 .ASCII <1>\-\ OOFC7 P.AHM: .BYTE 3, 0 .WORD 35 OOFC8 .BYTE 30, 30, 0, 0, 0, 0, 0 OOFD5 P.AHN: .BYTE 3, 0 .WORD 15 OOFD7 .BYTE 3, 0 .WORD 15 OOFD9 .BYTE 20, 20, 0, 0, 0, 0, 0 OOFE1 .ASCII <1>\=\ OOFE5 .BYTE 3, 0 .WORD 14 .OOFE5 .BYTE 3, 0 .WORD 14 .OOFE5 .BYTE 3, 0 .WORD 14 .OOFE5 .BYTE 3, 0 .WORD 14 .OOFF2 P.AHP: .BYTE 3, 0 .OOFF4 .BYTE 3, 0 .OOFF6 .BYTE 20, 20, 0, 0, 0, 0, 0
		00 00 00 1E 1E 26 01 00 03 0000 00 00 14 14	OOF CB
		3D 01	OOFD9
		3D 2F 02 00 03 0013 00 00 00 14 14	00FE7 .BYTE 20, 20, 0, 0, 0, 0, 0 00FEF .ASCII <2>\/=\ 00FF2 P.AHP: .BYTE 3, 0 00FF4 .WORD 19 00FF6 .BYTE 20, 20, 0, 0, 0, 0
		00 03 0015	OOFFE
	00 00 00	00 03 000F	0100C
	00 00 00	00 00 00 14 14 3E 01 00 03 0011 00 00 00 14 14 3D 3E 02	01016 P.AHS: .BYTE 3, 0 0101F .WORD 17 01021 .BYTE 20, 20, 0, 0, 0, 0
00000ED3 00000EC5 00000F28 00000F1A 00000F7D 00000F6F	00000EB7 00000EA9 00000F0C 00000EFE 00000F60 00000F52	00 00 00 14 14 3D 3E 02 00000014 00000E9B 00000E8D 00000EF0 00000EE1 00000F44 00000F36 00000F9A 00000F8C	01029 .ASCII <2>\>=\ 0102C .LONG 20 01030 P.AGY: .LONG 3725, 3739, 3753, 3767, 3781, 3795, 3809, - 01048 3824, 3838, 3852, 3866, 3880, 3894, 3908, - 01060 3922, 3936, 3951, 3965, 3980, 3994
		00000F9A 00000F8C 01 04 0000B 00 00 00 00 00 27 01	01078 01080 P.AHT: .BYTE 4, 1 01082 .WORD 11 01084 .BYTE 0, 0, 0, 0, 0, 0, 0

*

DBGPARSER V04-000												1	3 5-Sep-19 4-Sep-19	084 02:10 084 12:17	:13	VAX-11 Bliss-32 V4.0-742 [DEBUG.SRCJDBGPARSER.B32;1	Page 80 (17)
	44	45	4E	49	41 00 00 00 00 00	52 00 00 48 00 00 00	54 00 54 00 00 00	53 547 445 43	4555455454C	4F 441 4545 4545 4545 4545	43 OB 46 O5 40 O6 50 O3 55 O3 55 O3 55 O3 56 O3	01090 01090 010A4 010AC 010B4 010C0 010C8 010D0 010D4	P.AHU: P.AHV: P.AHX: P.AHZ: P.AIA: P.AIB: P.AIC: ADA_TIC	ASCII	<11><5>\ <4>\ <6>\ <3>\ <4>\ <4>\ <4>\ <3>\	\CONSTRAINED\ \FIRST\<0><0> \LAST\<0><0> \LAST\<0><0> \LENGTH\<0> \POS\ \PRED\<0><0><0> \SIZE\<0><0><0> \SUCC\<0><0> \VAL\	***************************************
00001035	00001	031		0102	0	0001			00	00 00 00 2E 00	000100D 000103D 0000103D 000000 000000 000000 000000 0000000 0000	010FC 0110F 01100 01108 01108 01108 01108 01110 011118 01118 01118 01118 01118 01118 01118 01118 01118 01118 01118	P.AIF: P.AIG:	BYTE WORD BYTE WORD BYTE ASCII BYTE WORD BYTE ASCII BYTE WORD BYTE ASCII BYTE LONG LONG LONG	0.0000000000000000000000000000000000000		-

DBGPARSER	16-Sep	-1984 02:10:13	VAX-11 Bliss-32 V4.0-742	Page 81
V04-000	14-Sep	-1984 12:17:30	EDEBUG.SRCJDBGPARSER.B32;1	(17)
	00 01160 09 01161 003E 01162 01 01165 0002 01166 00A 01169 003E 0116A 01 0116C 01 0116C 01 01170 012 01172 03 01174 0A 01175 003E 01176 08 01178 007 01170 0014 0117E 00 01180 12 01181 0014 01182 09 01184 007 01185 0014 01186 00 01188 09 01188 09 01188 09 01188 09 01188 09 01188 00 01190 00A 01191 003E 01190 00A 01191 003E 01190 00A 01191 003E 01190 00A 01191 003E 01190 00A 01191 001B 01190 00A 01191 00A 01191 00A 01190 00A 01191 00A 01190	BYTE 092 BYTE 092 BYTE 092 BYTE 100 BYTE 11 WORD 11 BYTE 12 BYTE 12 BYTE 12 BYTE 12 BYTE 13 BYTE 14 BYTE 14 BYTE 162 BYTE	EVEDUO. SINCIPOUT ANGEN. DJZ.; I	

DE

00

0021

33

DE

V

DI

DI

DBGPARSER V04-000								f 4 16-Sep-19 14-Sep-19		0:13 VAX-11 Bliss-32 V4.0-742 7:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 85
000010A9 00000000	000011BD	000010C1 00000001	0000	0FAD	000	00E71	0011 04 26 0016 01 2A 0010000 0000000 0000000 0000000 0000000	0129A 0129C 0129D 0129E 012AO 012A1 012A2 012A4 012A8 012AC P.AIM:	.WORD .BYTE .WORD .BYTE .WORD .LONG .LONG	17 38 22 1 42 26 0 3545, 3697, 4013, 4289, 4541, 4265, 4285, 4289, 0, 1, 0, 0, 0	
3E042800	3C04A800	5E010800		2800		00004	00000000 00000007 2E01803E 3D042800	012DC 012E0 012E4 P.AIN: 012FC	.LONG	7 771850302, 620756996, 973350912, - 1577125888, 1006938112, 1040459776, - 1023682560	
		C	00 00	00	00	00 54	00 02 001E 00 C8 2D 4F 4E 03 00 03	01300 P.AIP: 01302 01304 01300 01310 P.AIQ:	BYTE WORD BYTE ASCII BYTE WORD BYTE WORD BYTE WORD BYTE ASCII BYTE WORD BYTE BYTE BYTE	36 45, -56, 0, 0, 0, 0, 0 33\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	
		C	00 00	00	00	00	001F 00 28 28 4E 41 03 00 03	01312 01314 0131C 01320 P.AIR:	.WORD .BYTE .ASCII	31 40, 40, 0, 0, 0, 0, 0 3>\AND\ 3, 0	
		C	00 00	00	00	00	0020 00 1E 1E 52 4F 02	01322 01324 0132C 0132F P.AIS:	.WORD .BYTE .ASCII .BYTE	32 30, 30, 0, 0, 0, 0, 0 <2>\OR\ 3, 0	
		O	00 00	00	00	00	0021 00 1E 1E 4F 58 03 00 03 003A 00 14 14	01331 01333 01338	.WORD .BYTE .ASCII .BYTE	33 30, 30, 0, 0, 0, 0, 0 <3>\XOR\ 3, 0	
		O	00 00	00	00	00 50	00 14 14 40 49 03 00 03	0133F P.AIT: 01341 01343 0134B 0134F P.AIU:	.WORD .BYTE .ASCII .BYTE	58 20, 20, 0, 0, 0, 0, 0 3>\IMP\ 3, 0	
		O	00 00	00	00	00 56	4D 49 03 00 03 0022 00 0A 0A 51 45 03	01351 01353 0135B 0135F	.BYTE .ASCII	10, 10, 0, 0, 0, 0, 0 <3>\EQV\	
00001200	000012BC	000012AC	0000	129D	000	01280	00000006 00001270 01 02 0002	01360 01364 P.AIO: 0137C P.AIW: 0137E	.LONG .LONG .BYTE .WORD	4733, 4749, 4765, 4780, 4796, 4812 2, 1	
		(00 00	00	00	00	00 00 00 5C 01 01 03 0003	01380 01388 0138A P.AIX:	ASCII BYTE WORD BYTE ASCII BLKB LONG LONG BYTE WORD BYTE WORD BYTE	0, 0, 0, 0, 0, 0, 0 <1><92> 3, 1	
		(00 00	00	00	00	00 00 00 5C 01 01 03	0138E 01396 01398 P.AIY:	BYTE ASCII BYTE WORD BYTE	0, 0, 0, 0, 0, 0, 0 \$1><92> 3, 1	
		(00 00	00	00	00	00 00 00 3A 3A 02 01 04 0004	01390 01380 01384 01387 P.AIZ: 01389	BYTE ASCII BYTE WORD	0. 0. 0. 0. 0. 0 4. 1	

D

DBGPARSER V04-000			G 4 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 86
	00 00 00	00 00 00	00 00 013AB	
	00 00 00	00 00 00	000B 013B7 .WORD 11 C8 05 013B9 .BYTE 5, -56, 0, 0, 0, 0, 0 28 01 013C1 .ASCII <1>\(\)\(\)\(\) 02 04 013C3 P.AJB: .BYTE 4, 2 .WORD 12	
	00 00 00	00 00 00	000C 013C5 .WORD 12 06 C8 013C7 .BYTE -56, 6, 0, 0, 0, 0, 0, 0 29 01 013CF .ASCII <1>\)\ 00 02 013D1 P.AJC: .BYTE 2, 0 .WORD 4	
	00 00 00	00 00 00	0004 013D3 .WORD 4 C8 46 013D5 .BYTE 70, -56, 0, 0, 0, 0, 0 2B 01 013DD .ASCII <1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	
	00 00 00	00 00 00	0005 013E1 .WORD 5 C8 46 013E3 .BYTE 70, -56, 0, 0, 0, 0, 0 2D 01 013EB .ASCII <1>\n\ 00 03 013ED P.AJE: .BYTE 3, 0	
	00 00 00	00 00 00 2A	00 00 013AB	
	00 00 00	00 00 00	000A 013FE .WORD 10 5C 5A 01400 .BYTE 90, 92, 0, 0, 0, 0, 0 5E 01 01408 .ASCII <1>\^\ 00 03 0140A P.AJG: .BYTE 3, 0	
	00 00 00	00 00 00	0008 01400 .WORD 8 50 50 0140E .BYTE 80, 80, 0, 0, 0, 0, 0 2A 01 01416 .ASCII <1>*\ 00 03 01418 P.AJH: .BYTE 3, 0	
	00 00 00	00 00 00	00 03 01418 P.AJH: BYTE 3, 0 0009 0141A	
	00 00 00	00 00 00	00 03 01426 P.AJI: .BYTE 3, 0 0006 01428 .WORD 6 3C 3C 0142A .BYTE 60, 60, 0, 0, 0, 0, 0, 0 2B 01 01432 .ASCII <1>\+\ 00 03 01434 P.AJJ: .BYTE 3, 0	
	00 00 00	00 00 00	00 03 01434 P.AJJ: BYTE 3, 0 0007 01436 .WORD 7 3C 3C 01438 .BYTE 60, 60, 0, 0, 0, 0, 0, 0 2D 01 01440 .ASCII <1>\-\ 00 03 01442 P.AJK: BYTE 3, 0	
	00 00 00	00 00 00	00 03 01442 P.AJK: .BYTE 3, 0 0013 01444	
	00 00 00	00 00 00	00 03 01450 F.AJL: BYTE 3, 0 0015 01452 WORD 21 32 32 01454 BYTE 50, 50, 0, 0, 0, 0, 0, 0 3C 02 0145C ASCII <2>\<=\ 00 03 0145F P.AJM: BYTE 3, 0	
	00 00 00		32 32 01463 .BYTE 50, 50, 0, 0, 0, 0, 0, 0 30 02 0146B .ASCII <2>\=<\	
	00 00 00		00 03 0146E P.AJN: .BYTE 3, 0 000F 01470 .WORD 15 32 32 01472 .BYTE 50, 50, 0, 0, 0, 0, 0	

DBGPARSER V04-000			H 4 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 8
V04-000	00 00 00 00 00 00 00 00 00 00 00 00 00 00	3D 3E 02 00 00 32 32 00 03 00 00 32 32 00 00 1345 00 00 1345 00 00 00 00 00 00 00 00 00	0147A	

DV

DBGPARSER V04-000	I 4 16-Sep-1984 02:10:13 14-Sep-1984 12:17:30	VAX-11 Bliss-32 V4.0-742 Page 88 [DEBUG.SRC]DBGPARSER.B32;1 (17)
000014C9 000014DD 00000C8D 00001449 000012E1 00000000 00000000 00000001 00000000 000014DD 5E010800 5D040000 5B000800 3E040000 3C008800	000F 0156E	LDEBUG.SRCJDBGPARSER.B32;1 (17)
	00 03 01614 P.AKD: .BYTE 3. 0	

0011

0014

00

00

00

00

00

00

00

00

00

00

00

00

00

016AB

016AD 016B5

016BC

016BE 016C6

016BA P.AKN:

016CA P.AKO: 016CC 016CE 016D6

016DB P.AKP: 016DD

016DD 016DF 016E7 016EC P.AKQ: 016FE 016FO 016F8 016FC P.AKR: 016FE

3. 0

3, 0

3, 0 18

3, 0

3₀ 0

50, 50, 0, 0, 0, 0, 0, 0 4>\GTRA\

50, 50, 0, 0, 0, 0, 0, 0 <3>\GEQ\

50, 50, 0, 0, 0, 0, 0, 0 44>\GEQU\

50, 50, 0, 0, 0, 0, 0, 0 4>\GEQA\

\$6, 50, 0, 0, 0, 0, 0, 0 \$3>\LS\$\

.BYTE

. WORD

.BYTE

.BYTE

. WORD

.BYTE .ASCII

.BYTE . WORD

.BYTE

.BYTE . WORD

.BYTE

.ASCII

.BYTE .WORD .BYTE .ASCII

.BYTE

.ASCII

.ASCII

DBGPARSER V04-000		K 4 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 Page 90 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1 (17)
00 00	00 00 00 00 32 32 55 53 53 4C 04	2 01700 .BYTE 50, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
00 00	00 00 00 00 32 32 41 53 53 4C 04	3 0170D P.AKS: .BYTE 3, 0 4 0170F .WORD 20 2 01711 .BYTE 50, 50, 0, 0, 0, 0, 0 4 01719 .ASCII <4>\LSSA\ 3 0171E P.AKT: .BYTE 3, 0
00 00	00 00 00 00 32 32 51 45 4C 03	. WORD 21 2 01722 .BYTE 50, 50, 0, 0, 0, 0, 0 3 0172A .ASCII <3>\LEQ\ 3 0172E P.AKU: .BYTE 3, 0
00 00	00 00 00 00 32 32 00 00 00 00 32 32 00 03 00 03 00 03 00 03 00 03 00 03	3 0172E P.AKU: .BYTE 3, 0 5 01730 .WORD 22 01732 .BYTE 50, 50, 0, 0, 0, 0, 0 4 0173A .ASCII <4>\LEQU\ 3 0173F P.AKV: .BYTE 3, 0
00 00	00 00 00 00 32 32 41 51 45 4C 04	01741 .WORD 22 01743 .BYTE 50, 50, 0, 0, 0, 0, 0 4 0174B .ASCII <4>\LEGA\ 2 01750 P.AKW: .BYTE 2, 0
00 00	00 00 00 00 32 32 32 32 32 32 32 32 32 32 32 32 32	2 01750 P.AKW: .BYTE 2, 0 E 01752 .WORD 30 B 01754 .BYTE 40, -56, 0, 0, 0, 0, 0 3 0175C .ASCII <3>\NOT\ 3 01760 P.AKX: .BYTE 3, 0
00 00	00 00 00 00 1E 1E 44 4E 41 03 00 03	E 01764 .BYTE 30, 30, 0, 0, 0, 0, 0 ; 3 0176C .ASCII <3>\AND\
00 00	00 00 00 00 14 14 52 4F 02 00 03	3 01770 P.AKY: .BYTE 3, 0 0 01772 .WORD 32 4 01774 .BYTE 20, 20, 0, 0, 0, 0, 0 2 0177C .ASCII <2>\OR\ 3 0177F P.AKZ: .BYTE 3, 0
00 00	00 00 00 00 0A 0A 0A 0A 0A 0A 0A 0A 0A 0	3 0177F P.AKZ: .BYTE 3, 0 2 01781 .WORD 34 3 01783 .BYTE 10, 10, 0, 0, 0, 0, 0 3 0178B .ASCII <3>\EQV\ 3 0178F P.ALA: .BYTE 3, 0
00 00	00 00 00 00 0A 0A 0A 52 4F 58 03	3 0178F P.ALA: .BYTE 3, 0 1 01791 .WORD 33 A 01793 .BYTE 10, 10, 0, 0, 0, 0, 0 3 0179B .ASCII <3>\XOR\ 0179F .BLKB 1
000015E3 000015D3 000015C2 00001 00001647 00001637 00001626 00001 000016AB 0000169B 0000168A 00001 0000170C 000016FC 000016ED 00001	01 02	5637, 5653, 5670, 5687, 5703, 5720, 5737, -: 8 01704 5753, 5770, 5787, 5803, 5820, 5837, 5853, -: 9 017EC 5869, 5884, 5900 ::
00 00	00 00 00 00 00 00 00 00 00 00 00 00 00	0 01806 .WORD 2 0 01808 .BYTE 0, 0, 0, 0, 0, 0, 0 1 01810 .ASCII <1><92>
00 00	00 00 00 00 00 00 00 5C 01 01 04	3 01814 .WORD 3 0 01816 .BYTE 0, 0, 0, 0, 0, 0, 0 1 0181E .ASCII <1><92> 4 01820 P.ALE: BYTE 4, 1
00 00	00 00 00 00 00 00 00 00 00 00 00 00 00	1 0181E

DBGPARSER V04-000	16-Sep-1984 02:10:13 14-Sep-1984 12:17:30	VAX-11 Bliss-32 V4.0-742 Page 91 (17)
00 00 00 0	00 00 00 C8 05 01830 .WORD 11 28 01 0183A .ASCII <1 02 04 0183C P.ALG: .BYTE 4	-56. 0. 0. 0. 0. 0
00 00 00 0	00 00 00 06 C8 01840 .BYTE -5 29 01 01848 .ASCII <1 02 04 0184A P.ALH: .BYTE 4	6, 6, 0, 0, 0, 0, 0
00 00 00 0	00 02 01858 P.ALI: BYTE 2	6, 110, 0, 0, 0, 0, 0
00 00 00 0	ON ON ON THE ALL DIRECT ON ON ON	00, -56, 0, 0, 0, 0, 0
00 00 00 0	00 00 00 C8 5A 0186A .BYTE 90 2B 01 01872 .ASCII <1 00 02 01874 P.ALK: .BYTE 2, 0005 01876 .WORD 5), -56, 0, 0, 0, 0, 0 >\+\
	00 00 00 00 08 5A 01878 .WORD 5 2D 01 01880 .ASCII <1 00 03 01882 P.ALL: BYTE 3 0026 01884 .WORD 38	0, -56, 0, 0, 0, 0, 0 1>\-\ 3
	00 03 01890 P.ALM: .BYTE 3, 0008 01892 .WORD 8	
	00 00 00 46 46 01894 BYTE 70 ASCII <1 00 03 0189E P.ALN: BYTE 3. WORD 9 00 00 00 46 46 018A2 BYTE 70	
	00 00 00 46 46 018A2 .BYTE 70 2F 01 018AA .ASCII <1 00 03 018AC P.ALO: .BYTE 3. 0006 018AE .WORD 6 00 00 00 3C 3C 018B0 .BYTE 60 2B 01 018B8 .ASCII <1 00 03 018BA P.ALP: .BYTE 3. 0007 018BC .WORD 7), 70, 0, 0, 0, 0, 0 >\/\ , 0
	00 00 00 46 46 018A2 .BYTE 70 2F 01 018AA .ASCII <1 00 03 018AC P.ALO: .BYTE 3, 0006 018AE .WORD 6 00 00 00 3C 3C 018B0 .BYTE 60 2B 01 018B8 .ASCII <1 00 03 018BA P.ALP: .BYTE 3, 00 07 018BC .WORD 7 00 00 00 3C 3C 018BE .BYTE 60 2D 01 018C6 .ASCII <1 0000000E 018C8 .LONG 14	
	000017F3 000017D5 018F4 61	0, 60, 0, 0, 0, 0, 0 017, 6031, 6045, 6059, 6073, 6087, 6101, - 115, 6129, 6143, 6157, 6171, 6185, 6199
	00001837 00001829 018FC 00 00 01904 P.ALR: .BYTE C. 0002 01906 .WORD 2 00 00 00 01908 .BYTE O. 50 01 0190C .ASCII <1 02 00 0190E P.ALS: .BYTE O.	0 >01 >01 >01 >01
	00 00 01904 P.ALR: .BYTE C. 0002 01906 .WORD 2 00 00 00 01908 .BYTE 0. 5D 01 0190C .ASCII <1 02 00 0190E P.ALS: .BYTE 0. 0003 01910 .WORD 3 00 00 00 01912 .BYTE 0. 3A 01 01916 .ASCII <1 00 00 01918 P.ALT: .BYTE 0. 0001 0191A .WORD 1	2°, 0°, 0°, 0°, 0°, 0°, 0°, 0°, 0°, 0°, 0
	00 00 01918 P.ALT: BYTE 0.	. 0

DBGPARSER V04-000	M 4 16-Sep-1984 02:10:13 VA) 14-Sep-1984 12:17:30 EDE	(-11 Bliss-32 V4.0-742 Page 92 BUG.SRCJDBGPARSER.B32;1 (17)
00	00 00 00 0191C .BYTE 0, 0, 0, 0, 20 ASCII \$1>\\	0 :
00001895 0000188	00000000 01934 .LONG 0 01938 P.ALU: .BLKB 0 00000000 01938 .LONG 0	33, 6293
	0193C P.ALV: .BLKB 0 02 0193C P.ALW: .BYTE 2 01 0193D .BYTE 1 0006 0193E .WORD 6 03 01940 .BYTE 3	
	0006 0193E .WORD 6 03 01940 .BYTE 3 03 01941 .BYTE 3 0008 01942 .WORD 8 09 01944 .BYTE 9 0F 01945 .BYTE 15 0008 01946 .WORD 8	
	0008 01946 .WORD 8 04 01948 .BYTE 4 08 01949 .BYTE 8 000D 0194A .WORD 13 01 0194C .BYTE 1 0C 0194D .BYTE 12 000F 0194E .WORD 15	
	000F 0194E .WORD 15 00000000 01950 .LONG 0 01 01954 .BYTE 1 02 01955 .BYTE 2 000F 01956 .WORD 15	
	000F 01956 .WORD 15 00000000 01958 .LONG 0 03 0195C .BYTE 3 0E 0195D .BYTE 14 0008 0195E .WORD 8	
	0E 01950 .BYTE 14 0008 0195E .WORD 8 09 01960 .BYTE 9 0F 01961 .BYTE 15	
	0008 01962 .WORD 8 04 01964 .BYTE 4 14 01965 .BYTE 20 0000 01966 .WORD 13	
	01 01968 .BYTE 1 17 01969 .BYTE 23 000F 0196A .WORD 15 00000000 0196C .LONG 0	
	00000000 0196C .LONG 0 01 01970 .BYTE 1 2A 01971 .BYTE 42 000F 01972 .WORD 15	
000018A5 000018B9 00000C8D 00001849 0000172 00000000 00000000 00000001 00000000 000018B	00001575 0197C P.ALX: .LONG 5493, 597 000018B5 01994 6329, 0,	21, 6217, 3213, 6329, 6309, 6325, -
3C00A800 25000800 2A008800 21002800 5F00000 7C002800 5E010800 5D040000 5B000800 3E04280	0000000D 019B0 2400007 019B4 P.ALY: LONG 603979783 3D042800 019CC 76000800 019E4 102368256 156054323 00 02 019E8 P.AMA: BYTE 2, 0	3. 1593835527. 553658368 3. 620759040. 1006675968 50. 1040459776. 1526728704 52. 1577125888. 2080385024
	00 02 019E8 P.AMA: .BYTE 2, 0	:

							N 4 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRCJDBGPARSER.B32;1	
00	90 46	00 4F	00 45	00 5A	99	0029 C8 8C 53 06	019EA .WORD 41 019EC .BYTE -116, -56, 0, 0, 0, 0, 0 019F4 .ASCII <6>\SIZEOF\ 019FB .BLKB 1 019FC .LONG 1	
00	00	00	00	00	00	0001965 01 02 0002 00 00	01A00 P.ALZ: .LONG 6501 01A04 P.AMC: .BYTE 2, 1 01A06 .WORD 2 01A08 .BYTE 0, 0, 0, 0, 0, 0	
00	00	00	00	00	00	5C 01 01 03 0003 00 00 5C 01 01 04	01A10 01A12 P.AMD: .BYTE 3, 1 01A14 .WORD 3 01A16 .BYTE 0, 0, 0, 0, 0, 0, 0 01A1E .ASCII <1><92>	
00	00	00	00	00	00	01 04 0004 00 00 5B 01 01 03	01A20 P.AME: .BYTE 4, 1 01A22 .WORD 4 01A24 .BYTE 0, 0, 0, 0, 0, 0, 0	
00	00	00	00	00	00	0005 00 00 2E 01	01A2E P.AMF: .BYTE 3, 1 01A30 .WORD 5 01A32 .BYTE 0, 0, 0, 0, 0, 0, 0 01A3A .ASCII <1>\.\	
00	00	00	00	00	00	02 02 000B C8 05 28 01 02 04	01A3C P.AMG: .BYTE 2, 2 01A3E .WORD 11 01A40 .BYTE 5, -56, 0, 0, 0, 0, 0 01A48 .ASCII <1>\(\) 01A4A P.AMH: .BYTE 4, 2	
00	00	00	00	00	00	000C 06 C8 29 01 00 02 0017	01A4A P.AMH: .BYTE 4, 2 01A4C .WORD 12 01A4E .BYTE -56, 6, 0, 0, 0, 0, 0, 0 01A56 .ASCII <1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	
00	00	00	00	00	00	0017 C8 8C 21 01 00 02	01A5C .BYTE -116, -56, 0, 0, 0, 0, 0 01A64 .ASCII <1>\!\ 01A66 P.AMJ: .BYTE 2, 0	
00	00	00	00	00	00	001E C8 8C 7E 01 00 02 0003	01A68 .WORD 30 01A6A .BYTE -116, -56, 0, 0, 0, 0, 0 01A72 .ASCII <1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	
00	00	00	00	00	00	C8 8C 2A 01 00 03 0008	01A78	
00	00	00	00	00	00	82 82 2A 01 00 03 0009	01A86	
00	00	00	00	00	00	82 82 2F 01 00 03 0025	01A94	
00	00	00	00	00	00	82 82 25 01 00 03 0026	01AA2	
00	00	00	00	00	00	6E 6E	01ABO .BYTE 110, 110, 0, 0, 0, 0, 0	

Page 93 (17)

DBGPARSER V04-000	B 5 16-Sep-1984 02:10 14-Sep-1984 12:17):13 VAX-11 Bliss-32 V4.0-742 Page 94 2:30 [DEBUG.SRCJDBGPARSER.B32;1 (17)
00 00 00	3C 3C 02 01AB8	<2>\<<\ 3,0 36 110, 110, 0, 0, 0, 0, 0 <2>\>>\
00 00 00	00 03 01ACA P.AMQ: .BYTE .WORD .WORD .BYTE .ASCII 00 03 01AD6 .ASCII 00 03 01AD8 P.AMR: .BYTE	36 0 100, 100, 0, 0, 0, 0, 0 100, 100, 0, 0, 0, 0
00 00 00	00 03 01AD8 P.AMR: .BYTE .WORD .WORD .BYTE .ASCII 00 03 01AE7 P.AMS: .BYTE	31 100, 100, 0, 0, 0, 0, 0 <2>\<=\ 3, 0
00 00 00	00 00 00 64 64 01AE9 .WORD .BYTE .ASCII 00 03 01AF5 P.AMT: .BYTE	15 100, 100, 0, 0, 0, 0, 0
00 00 00	00 00 00 64 64 01AF7 .WORD 00 00 64 64 01AF9 .BYTE 3D 3E 02 01B01 .ASCII 00 03 01B04 P.AMU: .BYTE	3, 0 17 100, 100, 0, 0, 0, 0, 0 <2>\>=\ 3, 0
00 00 00	00 00 00 6E 6E 01ABF ASCII 00 03 01ACA P.AMQ: BYTE 00 03 01ACA P.AMQ: BYTE 00 03 01ACA P.AMQ: BYTE 00 03 01ADA P.AMR: BYTE 00 03 01AEA P.AMS: BYTE 00 01 01AEA BYTE 00 03 01AEA P.AMS: BYTE 00 03 01AEA P.AMS: BYTE 00 01 01AEA BYTE 00 03 01AEA P.AMS: BYTE 00 00 01 01AEA BYTE 00 03 01AEA P.AMS: BYTE 00 03 01AEA P.AMS: BYTE 00 01 01AEA BYTE 00 01AEA BYTE 00 01 01AEA BYTE 00 01 01AEA BYTE 00 01 01AEA BYTE 00 01AEA BYTE 00 01 01AEA BYTE 01	3, 0 13 90, 90, 0, 0, 0, 0, 0 <2>\==\ 3, 0
00 00 00	00 00 00 5A 5A 01B17 .BYTE .ASCII 00 03 01B22 P.AMW: .BYTE	14 90, 90, 0, 0, 0, 0, 0 <2>\!=\ 3, 0
00 00 00	OO OZ OIDZO D AMV. DVTE	70, 70, 0, 0, 0, 0, 0 <1>\^\ 3, 0
00 00 00	00 03 01B30 P.AMX: BYTE 0020 01B32 .WORD 00 00 00 3C 3C 01B34 .BYTE 7C 01 01B3C .ASCII 00 03 01B3E P.AMY: BYTE 001D 01B40 .WORD 00 00 00 28 28 01B42 .BYTE 7C 7C 02 01B4A .ASCII 01B4D .BKB 0000198F 00001981 01B54 P.AMB: LONG	32 60, 60, 0, 0, 0, 0, 0 51>\!\ 3,0
00 00 00	00 00 00 28 28 01842 BYTE 70 70 02 0184A ASCII 0184D BLKB	40, 40, 0, 0, 0, 0, 0, 0 <2>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
000019C7 000019B9 000019AB 0000199D 00001A1B 00001A0D 000019FF 000019F1 00001A72 00001A64 00001A55 00001A47 00001ABB 00001AAD 00001A9F	00001A38 00001A29 01B84	6529. 6543. 6557. 6571. 6585. 6599. 6613 6627. 6641. 6655. 6669. 6683. 6697. 6712 6727. 6741. 6756. 6770. 6785. 6800. 6815 6829. 6843
00 00 00	7C 7C 02 0184A .ASCII 0184D .BLKB 00000017 01850 .LONG 0000198F 00001981 01854 P.AMB: .LONG 000019E3 000019D5 0186C 00001A38 00001A29 01884 00001A90 00001A81 0189C 00 02 01880 P.AMZ: .BYTE 0028 01882 .WORD 00 00 00 C8 8C 01884 .BYTE 26 01 0188C .ASCII 00 03 0188E P.ANA: .BYTE 001F 018C0 .WORD 00 00 00 50 50 018C2 .BYTE	-116, -56, 0, 0, 0, 0, 0 <1>\&\
00 00 00	00 02 01BB0 P.AMZ: .BYTE .WORD 00 00 00 08 8C 01BB4 .BYTE .ASCII 00 03 01BBE P.ANA: .BYTE .WORD 00 00 00 50 50 01BC2 .WORD 00 03 01BC2 .BYTE .ASCII 00 03 01BC2 .BYTE .ASCII 00 03 01BC2 .BYTE .ASCII 00 03 01BCC P.ANB: .BYTE .WORD	31 80, 80, 0, 0, 0, 0, 0
	OOTC OTBCE PAND WORD	36 0

DE

000

0016

80 A0 01C6A

01C6C 01C6D 22

10

. WORD

.BYTE

.BYTE

	DBGPARSER V04-000						D 5 16-Sep-19 14-Sep-19	84 02:10 84 12:17):13 :30	VAX-11 BL EDEBUG.SR	iss-32 V4	.0-742 ER.B32;1	Pa	ge 96 (17)
						0011 01 00 001B 00000000	01C6E 01C70 01C71 01C72 01C74		17 1 12 27 0					
						01 02 001B 00000000	01C78 01C79 01C7A 01C7C 01C80 01C81	.BYTE .BYTE .WORD .LONG .BYTE	1 27 0 3					
						03 0E 000A 09 0F 000A 05	01C82 01C84 01C85 01C86 01C88 01C89	.WORD .BYTE .BYTE .WORD .BYTE	14 10 9 15 10 5					
						0011 04 12 0016 08	01C8A 01C8C 01C8D 01C8E 01C90	WOTE DO STEED SETED SETE	16 17 4 18 22 8					
The second secon						0011 01 17 001B 00000000	01C91 01C92 01C94 01C95 01C96 01C98 01C9C	.WORD .BYTE .BYTE .WORD .LONG	21 17 1 23 27					
The second second second second second						05 19 0011 04 10 0016	01C9D 01C9E 01CA0 01CA1 01CA2	BYTE WORD BYTE BYTE WORD	25 17 4 28 22					
						08 1F 0011 01 21	01CA4 01CA5 01CA6 01CA8 01CA9	BYTE BYTE WORD BYTE BYTE	8 31 17 1 33 27					
						001B 00000000 05 24 0011 04	01CBC 01CB0 01CB1 01CB2 01CB4 01CB5	.LONG .BYTE .BYTE .WORD .BYTE	0 5 36 17					
						0016 0016 08 28 0011 01 2A	01CA4 01CA5 01CA6 01CA9 01CAC 01CB0 01CB1 01CB2 01CB4 01CB5 01CB6 01CB8 01CB9 01CBA 01CBC 01CBD 01CBE 01CCO 01CCA 01CCBC	BYTE BYTE WORD BYTE BYTE WORD LONG BYTE WORD BYTE WORD BYTE WORD BYTE LONG LONG LONG	38 22 8 40 17 1					
	00001BC1 00000000	00001BD5 00000001	0000008D 00000000	00001AD1 00000001	0000197D 00001BD5	0018 0000000 0000000 00001931 00001801	01CBE 01CC0 01CC4 01CC8 P.ANO: 01CE0	.WORD .LONG .LONG	27 0 0 6449 7125	6525, 686 1, 0, 1,	5. 3213. 0. 0	7125, 7105,	7121, -	

DV

5

DBGPARSER V04-000								E 5 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRCJDBGPARSER.B32;1	Page 97 (17)
3400021F 3C04A800	3300021F 3900021F	3200021F 3800021F 24000007	310002 370002 5F0000	21F 21F 007	3000 3600 3004	0021F 0021F 02800	00000000 0000010 2008852 3500021F 3E042800	01CFE 01CFC 01D00 P.ANP: .LONG 755009618, 805306911, 822084127, - 01D18 01D30 838861343, 855638559, 872415775, - 889192991, 905970207, 922747423, - 939524639, 956301855, 1006938112, - 1040459776, 1023682560, 1593835527, -	
		00	00	00	00	00	01 03 0005 00 00 00 46 4F 02 01 03	01040 P.ANK: .BTTE 3, 1	
		00	00	00	00	00	0005 00 00 00 4E 49 02 00 02 0017	01D44	
		00	00	00	00	00 54	4F 4E 03	01D60 .WORD 23 01D62 .BYTE 25, -56, 0, 0, 0, 0, 0 01D6A .ASCII <3>\NOT\ 01D6E P.ANU: .BYTE 3, 0	
		00	00	00	00	00	0018 00 14 14 4E 41 03 00 03 0019	01D6E P.ANU: .BYTE 3, 0 01D70 .WORD 24 01D72 .BYTE 20, 20, 0, 0, 0, 0, 0 01D7A .ASCII <3>\AND\ 01D7E P.ANV: .BYTE 3, 0 01D80 .WORD 25	
		00	00	00	00	00	52 4F 02 02 03	01D80 .WORD 25 01D82 .BYTE 10, 10, 0, 0, 0, 0, 0 01D8A .ASCII <2>\OR\ 01D8D P.ANW: .BYTE 3, 2 01D8F .WORD 44	
		00	00	00	00	00 54	002C 00 1E 1E 4F 4E 03	01D8F .WORD 44 01D91 .BYTE 30, 30, 0, 0, 0, 0, 0 01D99 .ASCII <3>\NOT\ 01D9D .BLKB 3	
00001D0A	00001CFB	00001CEB	000010	CDB	0000	1000	00000006 00001CBD 00 03	01D9D .BLKB 3 01DAO .LONG 6 01DA4 P.ANQ: .LONG 7357, 7372, 7387, 7403, 7419, 7434 01DBC P.ANX: .BYTE 3, 0 01DBE .WORD 14 01DCO .BYTE 30, 30, 0, 0, 0, 0, 0	
		00	00	00 3D	20	00 54	000E 00 1E 1E 4F 4E 05 00 03	01DC0	
		00	00	00 3E	00 20	00 54	0015 00 1E 1E 4F 4E 05 00 03 0011	01DD0 .WORD 21 01DD2 .BYTE 30, 30, 0, 0, 0, 0, 0 01DDA .ASCII <5>\NOT >\ 01DE0 P.ANZ: .BYTE 3, 0 01DE2 .WORD 17	
		00	00	00 30	00 20	00 54	4F 4E 05	01DE2 .WORD 17 01DE4 .BYTE 30, 30, 0, 0, 0, 0, 0 01DEC .ASCII <5>\NOT <\ 01DF2 P.AOB: .BYTE 2, 1	
		00	00	00	00	00	00 00 00 5C 01 01 03	01DC8 01DCE P.ANY: BYTE 3, 0 01DD0 .WORD 21 01DD2 .BYTE 30, 30, 0, 0, 0, 0, 0, 0 01DDA .ASCII <5>\NOT >\ 01DE0 P.ANZ: BYTE 3, 0 01DE2 .WORD 17 01DE4 .BYTE 30, 30, 0, 0, 0, 0, 0, 0 01DEC .ASCII <5>\NOT <\ 01DF2 P.AOB: BYTE 2, 1 01DF4 .WORD 2 01DF6 .BYTE 0, 0, 0, 0, 0, 0, 0 01DFE .ASCII <1><92> 01E00 P.AOC: BYTE 3, 1 01E02 .WORD 3	
		00	00	00	00	00	00 00 00 50 01 01 04 0004	01E02 .WORD 3 01E04 .BYTE 0, 0, 0, 0, 0, 0, 0 01E0C .ASCII <1><92> 01E0E P.AOD: .BYTE 4, 1 01E10 .WORD 4	
							0004	OTETO .WORD 4	

P.AOR:

.BYTE

0000

00

00

00

00

00

0

30, 30, 0, 0, 0, 0, 0, 0

DBGPARSER V04-000				3D 01	G 5 16-Sep-1984 02:10: 14-Sep-1984 12:17: 01EDE ASCII		Page 99 (17)
00001E09 0	00001DFB 00001E4F	00001DED 00001E41	00001DBF 00001E33	00001D7D 00001D6F 00001DD1 00001DC3 000001E25 00001E17 00 00 00 00 00 00 00 00 00 00 00 00 00 00	01EE4 P.AOA: LONG 01F14 01F28 P.AOT: BYTE 01F30 01F32 P.AOU: BYTE 01F36 BYTE 01F40 BYTE 01F40 BYTE 01F40 BYTE 01F40 BYTE 01F60 P.AOX: BLKB 01F60 P.AOX: BLKB 01F60 P.AOX: BYTE 01F61 BYTE 01F62 BYTE 01F65 BYTE 01F65 BYTE 01F66 BYTE 01F67 BYTE 01F70 BYTE 01F70 BYTE 01F71 BYTE 01F72 WORD 01F74 BYTE 01F75 BYTE 01F76 BYTE 01F76 BYTE 01F76 BYTE 01F77 BYTE 01F77 BYTE 01F78 BYTE 01F78 BYTE 01F79 BYTE 01F79 BYTE 01F76 BYTE 01F77 BYTE 01F76 BYTE 01F77 BYTE 01F78 BYTE	<pre></pre>	

DBGPARSER V04-000								16-Sep-1 14-Sep-1	984 02:10: 984 12:17:	13 VAX-11 Bliss-32 V4.0-742 Pag 30 [DEBUG.SRC]DBGPARSER.B32;1	ge 10 (17
							0000	04 02034 27 02035 16 02036 01 02038 28 02039 18 0203A	.LONG .BYTE .BYTE .WORD	1 35 24 0 4 39 22 1 4 3 24	
00001EC9	00001F65 00000000	00001EDD 00000001	000	01E61 00000		01D2 01ED	0000 0000 0000 0000 0000	B 02041		43 24 0 0 7293, 7457, 7777, 7901, 8037, 7881, 7897, - 7901, 0, 1, 0, 1, 0 2 771817528, 788539392	
		(00 0	0 00	00	00	01 00 00 50	02090 P.APC: 02090 P.APE: 02092 0002094 0102096 03096 P.APF:	LONG LONG LONG BURB BYTE WORD BYTE	0. 0. 0. 0. 0. 0. 0 <1><92> 3. 1	
			00 0	0 00	00	00	00 00 50 01 00 00 28 01	00 020A2 01 020AA 04 020AC P.APG: 04 020AE	.BYTE .ASCII .BYTE .WORD .BYTE	0, 0, 0, 0, 0, 0, 0 <1><92> 4, 1 0, 0, 0, 0, 0, 0, 0	
				0 00	00	00	28 01 00 00 2E 00	02 02092 00 02094 01 0209C 03 0209E P.APF: 03 020A0 00 020A2 01 020AA 04 020AC P.APG: 04 020AE 00 020B0 01 020B8 03 020BA P.APH: 05 020BC 06 020BE 01 020C6 01 020C6 01 020C4 020C4	.ASCII .BYTE .WORD .BYTE .ASCII	0, 0, 0, 0, 0, 0, 0 3, 1 0, 0, 0, 0, 0, 0, 0 1>\.\	
		(00 0	0 00	00	00	00 3C	3 02006 P.APJ:	DUTE	3. 0 60, 60, 0, 0, 0, 0, 0 1>\+\ 3. 0	
		(00 0	00 00	00	00	00 3C	07 02008 8C 0200A 01 020E2 02 020E4 P.APK: 04 020E6 66 020E8	.MSCII	60, 60, 0, 0, 0, 0, 0 <1>\-\ 2, 0	
			00 0	00 00	00	00	00 C8	020E8 01 020F0 02 020F2 P.APL:	BYTE ASCII BYTE WORD	70, -56, 0, 0, 0, 0, 0 <1>\+\ 2, 0	
			00 0	00 00	00	00	00 C8	01 020F0 02 020F2 P.APL: 05 020F4 06 020F6 01 020FE 03 02100 P.APM: 08 02102	.ASCII	70, -56, 0, 0, 0, 0, 0 <1>\-\ 3, 0	

DBGPARSER V04-000			K 5 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 Page 10 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1 (1)
	00 00 00	00 00 00 50 50 2A 01 00 03	02104 .BYTE 80, 80, 0, 0, 0, 0, 0 0 0 10 10 10 10 10 10 10 10 10 10 10
	00 00 00	00 00 00 50 50 2F 01 00 03	02110 .WORD 9 02112 .BYTE 80, 80, 0, 0, 0, 0, 0 0211A .ASCII <1>\/\ 0211C P.APO: .BYTE 3, 0
	00 00 00	00 00 00 5C 5A 2A 2A 02 00 03	0211E .WORD 10 02120 .BYTE 90, 92, 0, 0, 0, 0, 0 02128 .ASCII <2>**\ 0212B P.APP: .BYTE 3, 0
	00 00 00	00 00 00 3C	0212D .WORD 35 0212F .BYTE 60, 60, 0, 0, 0, 0, 0 02137 .ASCII <2>\//\ 0213A P.APQ: .BYTE 2, 2
	00 00 00	00 00 00 C8 05 28 01 02 04	0213C .WORD 11 0213E .BYTE 5, -56, 0, 0, 0, 0, 0 02146 .ASCII <1>\(\) 02148 P.APR: .BYTE 4, 2
	00 00 00	00 00 00 06 C8	02104
00002053 00002045 000020A8 00002099	00002037 00002029 0000208B 0000207b	00002018 00002000 0000206F 00002061 000020C5 000020B7	02174 02186 8303, 8317, 8331, 8345, 8360, 8375, 8389
	00 00 00	00 00 00 C8 28 2E 01	02196 .WORD 3 02198 .BYTE 40, -56, 0, 0, 0, 0, 0
	00 00 00	0005	UZ1A4 .WUND 3
	00 00 00	00 00 00 32 32 2E 51 45 2E 04 00 03	021B2 .WORD 13 021B4 .BYTE 50, 50, 0, 0, 0, 0, 0 021BC .ASCII <4>\.EQ.\ 021C1 P.APW: .BYTE 3, 0
	00 00 00	00 00 00 32 32 00 00 00 32 32 00 03 00 00 00 32 32 2E 45 4E 2E 04 00 03 00 00 00 32 32 2E 54 47 2E 04 00 03	021C1 P.APW: .BYTE 3, 0 021C3 .WORD 14 021C5 .BYTE 50, 50, 0, 0, 0, 0, 0 021CD .ASCII <4>\.NE.\ 021D2 P.APX: .BYTE 3, 0
	00 00 00	00 00 00 32 32 2E 54 47 2E 04 00 03	021D2 P.APX: .BYTE 3, 0 021D4 .WORD 15 021D6 .BYTE 50, 50, 0, 0, 0, 0, 0
	00 00 00	00 00 00 32 32 2E 45 47 2E 04 00 03	021E3 P.APY: .BYTE 3, 0 021E5 .WORD 17 021E7 .BYTE 50, 50, 0, 0, 0, 0, 0
	00 00 00	00 00 00 32 32 2E 54 4C 2E 04 00 03	021A6

DBGPARSER V04-000									L 5 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 Page 1 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1
			00	00	00	00 2E	00 45	00 32 32 40 2E 04 00 02	02209
			00	00	00 2E	00 54	00 4F	00 C8 28 4E 2E 05 00 03	02218 0221A .BYTE 40, -56, 0, 0, 0, 0, 0 02222 .ASCII <5>\.NOT.\ 02228 P.AQC: .BYTE 3, 0
			00	00	00 2E	00	00 4E	00 1E 1E 41 2E 05 00 03	0222A .WORD 24 0222C .BYTE 30, 30, 0, 0, 0, 0, 0 0 02234 .ASCII <5>\.AND.\ 0223A P.AQD: .BYTE 3, 0
			00	00	00	00 2E	00 52	00 14 14 4F 2E 04 00 03	0223C .WORD 25 0223E .BYTE 20, 20, 0, 0, 0, 0, 0 02246 .ASCII <4>\.OR.\ 0224B P.AQE: .BYTE 3, 0 0224D .WORD 26 0224F .BYTE 10, 10, 0, 0, 0, 0, 0
			00	00	95 00	00 52	00 4F	001A	0224B P.AQE: .BYTE 3, 0 0224D .WORD 26 0224F .BYTE 10, 10, 0, 0, 0, 0, 0 02257 .ASCII <5>\.XOR.\ 0225D P.AQF: .BYTE 3, 0
			00	00	00 2E	00 56	00 51	00 0A 0A 45 2E 05 00 03	SYTE SO, O, O, O, O, O, O SO SO SO
			00	00 2E	00 56	00 51	00 45	00 0A 0A 4E 2E 06	02271 .WORD 26 02273 .BYTE 10, 10, 0, 0, 0, 0, 0 0227B .ASCII <6>\.NEQV.\ 02282 .BLKB 2
00002182 000021EC	00002171 000021DA	00002160 00002168	0 00	0002	14F 1B7	0000)213()21A	0000000C 0000212D 00002193 01 00	02288 P.APU: .LONG 8493, 8510, 8527, 8544, 8561, 8578, 8595, -; 022A0 8613, 8631, 8648, 8666, 8684 ;
							00	00 00 00 29 01 00 00	022BA .WORD 2 022BC .BYTE 0, 0, 0, 0 022CO .ASCII <1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
							00	00 00 00 2C 01 00 00	022C4 .WORD 1 022C6 .BYTE 0, 0, 0, 0 022CA .ASCII <1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
							00	00 00 00 00 00 00 3A 01	022CC P.AQK: .BYTE 0, 0 022CE .WORD 3 022DO .BYTE 0, 0, 0, 0
			0	0002	249	0000	02231	00000003 00002235 02 08 01 00000001	022D8
					45		52	02 08 01 00000000	022BB P.AQI: BYTE 0, 1 022BC
			2E	45	53	0000	41	00000002	022FF .ASCII <7>\.FALSE.\ 02307 .BLKB 1 02308 .LONG 2 0230C P.AQL: .LONG 8805, 8820
								00000000	02314 .LONG 0 : 02318 P.AQO: .BLKB 0 02318 P.AQP: .BYTE 2 :

DBGPARSER V04-000				M 5 16-Sep-1984 02:10:13 14-Sep-1984 12:17:30			3 VAX-1	1 Bliss-32 V	Page 10	
				01 0007 03 03 0009 09 06 0009	02319 0231A 0231C 0231D 0231E 02320 02321 02322	BYTE 1 WORD 7 BYTE 3 WORD 9 BYTE 9 BYTE 1	5			
				01 0007 03 0009 009 009 0009 0012 0012 0016 00000000	02319 02310 02310 02310 0231321 022333228 022333228 0223333335 0223333333 0223333333 0223333333 0223333333 02233333 02233333 02233333 02233333 02233333 02233333 02233333 0223333 0223333 0223333 0223333 0223333 022333 022333 022333 022333 022333 022333 022333 022333 02233	.BYTE 1	8			
				01 02 0016 0000000	02334 02335 02336 02338 02330 02330 0233E 02340	BYTE 1 BYTE 2 WORD 2 LONG 0 BYTE 3 BYTE 1				
				03 0E 0009 09 0F 0009 04 12 0012 01 17 0016 00000000	02341 02342 02344 02345 02346 02348 02349 0234A	.BYTE 1 .WORD 9 .BYTE 4 .BYTE 1 .WORD 1 .BYTE 1 .BYTE 2 .WORD 2	8 8 3 2			
				05 19 000E 04 10 0012 01 21	02350 02351 02352 02354 02355 02356 02358 02359	BYTE SOURCE SOUR	5 4 8 8			
				0016 00000000 05 24 000E 04 26 0012	0235A 0235C 02360 02361 02362 02364 02366	.WORD 2 .LONG 0 .BYTE 3 .WORD 1 .BYTE 4 .BYTE 3	64			
00002259 0000	2295 00000C8D	000020D9	00002000	01 2A 0016 0000000 0000000 00002001	02368 02369 0236A 0236C 02370 02374 P.AQQ:	BYTE 1 BYTE 4 WORD 2 LONG 0 LONG 8	2 2 193. 8205.	8409, 3213,	8853, 8793,	8841

.ASCII

DBGPARSER V04-000		B 6 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 107 (17)
	00 00 00 00 00 00 44 4E	00 03 02496 P.ARG: .BYTE 3, 0 001F 02498 .WORD 31 1E 1E 0249A .BYTE 30, 30, 0, 0, 0, 0, 0, 0 41 03 024A2 .ASCII <3>\AND\ 00 03 024A6 P.ARH: .BYTE 3, 0	
	00 00 00 00 00 00 52	0020 024A8 .WORD 32	
	00 00 00 00 00 00 56 51	0022 024B7 .WORD 34 0 0A 0A 024B9 .BYTE 10, 10, 0, 0, 0, 0, 0 1 45 03 024C1 .ASCII <3>\EQV\ 00 03 024C5 P.ARJ: .BYTE 3, 0 0021 024C7 .WORD 33	
	00 00 00 00 00 00 52 4F	45 03 024C1	
	00 00 00 00 00 00 44 4F	0 46 46 024D9	i
0000238F 0000237F 00 000023F2 000023E2 00 00002452 00002442 00	00236E 0000235E 0000234D 0023D1 000023C1 000023B0 002432 00002423 00002413	00002330 024FC P AGS: 10NG 9021 9037 9054 9070 9087 9103 9120	=
	00 00 00 00 00 00	00 00 02538 .BYTE 0, 0, 0, 0, 0, 0, 0 5C 01 02540 .ASCII <1><92> 01 03 02542 P.ARN: BYTE 3 1	
	00 00 00 00 00 00	00 00 02546 .WORD 3 00 00 02546 .BYTE 0, 0, 0, 0, 0, 0, 0 5C 01 0254E .ASCII <1><92> 02 02 02550 P.ARO: .BYTE 2, 2	
	00 00 00 00 00 00	nnnp nakka	
	00 00 00 00 00 00	29 01 0256A .ASCII <1>\)\ 02 04 0256C P.ARQ: .BYTE 4, 2	
	00 00 00 00 00 00	OU OF OFSILE LANG. "DITE E' O	
	00 00 00 00 00 00	0	
	00 00 00 00 00 00	40 01 02594 .ASCII <1>\a\	
	00 00 00 00 00 00	00 02 02596 P.ART: .BYTE 2, 0 0004 02598 .WORD 4 0 C8 5A 0259A .BYTE 90, -56, 0, 0, 0, 0, 0 2B 01 025A2 .ASCII <1>\+\	

DIV

00 02 025A4 P.ARU: .BYTE 2, 0 0005 025A6 00 00 00 00 00 00 00 08 5A 025A8 .BYTE 90, -56, 0, 0, 0, 0, 0 0 2D 01 025B0 .ASCII <1>-\ 00 03 025B2 P.ARV: .BYTE 3, 0 0026 025B4 .BYTE 80, 80, 0, 0, 0, 0, 0, 0 0 40 01 025BE .ASCII <1>\a\ 00 03 025C0 P.ARW: .BYTE 3, 0	DBGPARSER V04-000
00 00 00 00 00 00 00 00 00 00 00 00 00	

DV

DBGPARSER V04-000									D 6 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 P 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32:1	age 10
000025B5 00000000 5B000800	000025BD 00000000 2E058838	00000C8D 00000001 5E010800		2579 0000 2800		0246 025B	8	01 17 000B 0000000 0000000 00002329 000025B9 00000000 00000007 30008800	02664	
			0 00		00	00	00	00 03 0030 46 46 44 03 00 03	026C4 P.ASH: .BYTE 3, 0	
		O	0 00	00	00	00	00 4F	0024 46 46 40 03 00 03	02604 P.ASI: .BYIF 5. 0	
		O	00 00	00	00	00 4D	00 45	46 46 52 03 00 03	026E6 .WORD 37 026E8 .BYTE 70, 70, 0, 0, 0, 0, 0 026F0 .ASCII <3>\REM\ 026F4 P.ASK: .BYTE 3, 0	
			00 00		00	00	00 4E	46 46 41 03 00 03 0019	026F4 P.ASK: .BYTE 3, 0 026F6 .WORD 24 026F8 .BYTE 70, 70, 0, 0, 0, 0, 0 02700 .ASCII <3>\AND\ 02704 P.ASL: .BYTE 3, 0 02706 .WORD 25	
			0 00		00	00	52	3C 3C 4F 02 00 02 0017	02715 .WORD 23	
			0 00		00	00	00 4F	4E 03 00 03 002A	0271F .ASCII <3>\NOT\ 02723 P.ASN: .BYTE 3, 0 02725 .WORD 42	
00002690	00002681	00002671		2661		0265	90 4E	32 32 49 02 00000007 00002641 000026A0	02727	1
		(00 00	00	00	00	00	0002	02754 P.ASP: .BYTE 2, 1 02756 .WORD 2 02758 .BYTE 0, 0, 0, 0, 0, 0, 0 02760 .ASCII <1><92>	
		(00 00	00	00	00	00	0003	02760 .ASCII <1><92> 02762 P.ASQ: .BYTE 3, 1 02764 .WORD 3 02766 .BYTE 0, 0, 0, 0, 0, 0, 0 0276E .ASCII <1><92> 02770 P.ASR: .BYTE 4, 1	
		(00 00	00	00	00	00	00 00 5C 01 01 04 0004 00 00 5B 01 01 03	0276E	

D

						E 6 16-Sep-1984 02:10:13 VAX-11 BLiss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRCJDBGPARSER.B32;1	
00	00	00	00	00	00	0005 02780 .WORD 5 00 00 02782 .BYTE 0, 0, 0, 0, 0, 0, 0 2E 01 0278A .ASCII <1>\.\	
00	00	00	00	00	00	0007 0278E .WORD 7 00 00 02790 .BYTE 0, 0, 0, 0, 0, 0, 0 5E 01 02798 .ASCII <1>\^\	
00	00	00	00	00	00	000A 0279C .WORD 10 00 00 0279E .BYTE 0, 0, 0, 0, 0, 0, 0 28 01 027A6 .ASCII <1>\(\)	
00	00	00	00	00	00	000B 027AA .WORD 11 C8 05 027AC .BYTE 5, -56, 0, 0, 0, 0, 0 28 01 027B4 .ASCII <1>\(\)	
00	00	00	00	00	00	000C 02788 LIOPO 12	
00	00	00	00	00	00	5B 01 027D0 .ASCII <1>\[\] 00 02 027D2 P.ASY: .BYTE 2. 0	
00	00	00	00	00	00	0004 02704 .WORD 4 C8 3C 02706 .BYTE 60, -56, 0, 0, 0, 0, 0 2B 01 0270E .ASCII <1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	
00	00	00	00	00	00	C8 3C 027E4 .BYTE 60, -56, 0, 0, 0, 0, 0 2D 01 027EC .ASCII <1>\-\	
00	00	00	00	00	00 2A	50 50 027F2 .BYTE 80, 80, 0, 0, 0, 0, 0, 0 2A 02 027FA .ASCII <2>**\	
00	00	00	00	00	00	0008 027FF .WORD 8 46 46 02801 .BYTE 70, 70, 0, 0, 0, 0, 0 2A 01 02809 .ASCII <1>*\ 00 03 0280B P.ATC: .BYTE 3, 0 0009 0280D .WORD 9	
00	00	00	00	00	00	46 46 0280F .BYTE 70, 70, 0, 0, 0, 0, 0 0 2F 01 02817 .ASCII <1>\/\\ 00 03 02819 P.ATD: .BYTE 3, 0	
00	00	00	00	00	00	3C 3C 0281D .BYTE 60, 60, 0, 0, 0, 0, 0, 0 2B 01 02825 .ASCII <1>\+\ 00 03 02827 P.ATE: .BYTE 3, 0	
00	00	00	00	00	00	3C 3C 0282B .BYTE 60, 60, 0, 0, 0, 0, 0, 0 2D 01 02833 .ASCII <1>\-\ 00 03 02835 P.ATF: BYTE 3, 0	
00	00	00	00	00	00	0013 02837 .WORD 19 32 32 02839 .BYTE 50, 50, 0, 0, 0, 0, 0, 0 3C 01 02841 .ASCII <1>\lambda\leftarrow 00 03 02843 P.ATG: .BYTE 3, 0 0015 02845 .WORD 21	

Page 110 (17)

DBGPARSER V04-000								F 6 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 Pa 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	ge 111
		(00 00	00	00 0	0 00 3D	32 32 30 02 00 03	02847 .BYTE 50, 50, 0, 0, 0, 0, 0 0284F .ASCII <2>\<=\ 02852 P.ATH: .BYTE 3, 0	
		(00 00	00	00 0	0 00	32 32 3E 01 00 03	02854 .WORD 15 02856 .BYTE 50, 50, 0, 0, 0, 0, 0 0285E .ASCII <1>\>\ 02860 P.ATI: .BYTE 3, 0	
		(00 00	00	00 0	0 00 3D	32 32 3E 02 00 03	02862 .WORD 17 02864 .BYTE 50, 50, 0, 0, 0, 0, 0 0286C .ASCII <2>\>=\ 0286F P.ATJ: .BYTE 3, 0	
		(00 00	00	00 0	0 00	32 32 30 01	02847	
		(00 00	00	00 0	0 00 3E	32 32 30 02	0287F .WORD 14 02881 .BYTE 50, 50, 0, 0, 0, 0, 0 02889 .ASCII <2>\<>\	
00002717 0000276B 000027C0	00002709 00002750 00002782	000026FB 0000274F 000027A4 000027FA	0000 0000 0000 0000	26ED 2741 2796	00002 00002 00002	6DF 733 788	000E 32 32 3C 02 00000016 00002601 00002725 0000277A	0284F 02852 02854 02854 02856 02856 02856 02856 02856 02860 02860 02861 02861 02861 02867 02867 02867 02868 02868 02868 02868 02869 02878 02879 02881 02879 02880	
		00002774	0000	zrec		0 00	0002	028E8 P.ATM: .BYTE 0, 0 028EA .WORD 2 028EC .BYTE 0, 0, 0, 0 028FO .ASCII <1>\j\	
					0	0 00	00 00 50 01 02 00 0003 00 00 3A 01 00 00	028F2 P.ATN: .BYTE 0. 2	
					0	0 00	00 00 0001 00 00 20 01	UZOFE .WUND I	
			0000	2879	00002			02900	:
					0	0 02 5 52 0 02	00000001	02918 P.ATQ: .BYTE 1, 40, 2, 0 02910 .LONG 1 02920 .ASCII <4>\TRUE\ 02925 P.ATR: .BYTE 1, 40, 2, 0	
				45			54 04 28 01 00000000 46 05 00 01 000000000 4E 03	02900 02904 02906 02908 02908 02908 02908 02910 02910 02910 02920 02925 02925 02925 02927 02929 02929 02929 02929 02920 035CII <4>\TRUE\ 02933 02933 02937 02938 02937 02938 02937 02938 02940 02954 02955 02954 02955 02955 02956 02958 02958 02958 02959 02958 02959 02958 02959 02958 02959 02958 02959 02958 02959 02958 02958 02959 02958 02959 02958 02959 02958 02959 02958 02959 02958 02959 02958 02959 02958 02959 02958 02959 02958 02959 02958 02959	
			0000	2000		C 49	4E 03	0293B .ASCII <3>\NIL\ 0293F .BLKB 1 02940 .LONG 3	
			0000	2080	00002	OAZ	00000003 00002895 00000000	02944 P.ATP: .LONG 10389, 10402, 10416 02950 .LONG 0 02954 P.ATT: .BLKB 0 02954 P.ATU: .BYTE 2	
							02 01 0009 03 03	02954 P.ATU: .BYTE 2 02955 .BYTE 1 02956 .WORD 9 02958 .BYTE 3 02959 .BYTE 3	

	G 6 16-Sep 14-Sep	-1984 02:10:13 -1984 12:17:30	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page 112 (17)
000B	0295A 0295C	.WORD 11		•
000B 09 0F 000B 05 04 0012	0295D 0295E	BYTE 15		
05	02960 02961	BYTE 5 BYTE 4 WORD 18		
04	02962 02964 02965	BYTE 4		
0017	02966 02968	WORD 23		
07 0A 001C	02969 0296A	.BYTE 4 .BYTE 7 .WORD 23 .BYTE 7 .BYTE 10		
0A 0D 0021	0296C 0296D	BYTE 13		
01	0296E 02970 02971	BYTE 1		
00000000 0021 00000000	02972 02974	.WORD 33		
01	02978 02979	BYTE 1		
00000000	0297A 0297C	.WORD 33 .LONG 0 .BYTE 3 .BYTE 14		
03 0E 000B	02980 02981 02982	BYTE 14		
09 0F	02984 02985	BYTE 9		
000B 05	02986 02988	.WORD 11		
0012	02989 0298A	.BYTE 16		
04 12 0017	0298C 0298D 0298E	BYTE 4 BYTE 18 WORD 23 BYTE 7		
07 15 0010	02990 02991	BYTE 7		
001C 01 17	02992 02994	.BYTE 21 .WORD 28 .BYTE 1		
00000000 00000000	02995 02996	.BYTE 23		
00000000 05 19 0012	0298D 0298E 02990 02991 02992 02994 02995 02996 02996 02990 0299D 0299E 029A0	BYTE 5		
0012	0299E 029A0	BYTE 25		
04 10 0017	029A1 029A2	.BYTE 28		
07 1F	029A4 029A5 029A6	BYTE 7 BYTE 31 WORD 28		
001C	029A6 029A8	.WORD 28		
01 21 0021 0000000 05	029A8 029A9 029AA 029AC 029B0	BYTE 18 WORD 28 BYTE 21 WORD 28 BYTE 23 WORD 33 LONG 0 BYTE 28 WORD 18 BYTE 28 WORD 28 BYTE 28 WORD 28 BYTE 31 WORD 28 BYTE 31 WORD 33 LONG 0 BYTE 31		

DBGPARSER V04-000		H 6 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 Page 113 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1 (17)
	0012 04 0017 07 28 0010 01 2A 0021 00000000 05 20 0012 04 2D 0017 07 2E 0010 01 2F 0010 01 2F 0010 01 2F 00000000 00000000 00000000 00000000 0000	02981
00	00 00 00 00 00 00 00 00 00 00 00 00 00	02A44 P.ATX: .BLKB 0 02A44 P.ATZ: .BYTE 2, 1 02A46 .WORD 2 02A48 .BYTE 0, 0, 0, 0, 0, 0, 0 02A50 .ASCII <1><92>

00	00	00	00	00	00	000C 06 C8 29 01 00 02	02A8C 02A8E 02A96 02A98 P.AUF:	.WORD .BYTE .ASCII .BYTE	12 -56, 6, 0, 0, 0, 0, 0, 0 <1>()\ 2, 0
00	00	00	00	00	00	C8 46 2B 01	02A9A 02A9C 02AA4	.WORD .BYTE .ASCII	70, -56, 0, 0, 0, 0, 0, 0
						00 02	02AA6 P.AUG:	.BYTE	2. 0
00	00	00	00	00	00	C8 46	8AASO	.WORD	
00	00	00	00	00	00	C8 46 20 01 00 02	02AB2	.ASCII	70, -56, 0, 0, 0, 0, 0, 0
						00 02	02AB4 P.AUH:	BYTE	260
00	00	00	00	00	00	001E	02AB6	.WORD	
00	00	00	00	00	00	C8 1E 5E 01	02AB8 02AC0	.BYTE	30, -56, 0, 0, 0, 0, 0, 0
						C8 1E 5E 01 00 03	OZACZ P.AUI:	BYTE	360
						000A	02AC4	.WORD	
00	00	00	00	00	00 2A	5C 5A	02AC6	.BYTE	90, 92, 0, 0, 0, 0, 0, 0
					CM	5C 5A 2A 02 00 03	02ACE 02AD1 P.AUJ:	.ASCII	3, 0
						8000	02AD3	.WORD	8
00	00	00	00	00	00	50 50	02AD5	.BYTE	80, 80, 0, 0, 0, 0, 0, 0
						2A 01 00 03	02ADD 02ADF P.AUK:	.ASCII	3, 0
						0009	02AE1	.WORD	3. 0
00	00	00	00	00	00	50 50	02AE3	.BYTE	80, 80, 0, 0, 0, 0, 0, 0
						2F 01	02AEB 02AED P.AUL:	.ASCII	3, 0
						0006	OZAEF P.AUL.	WORD	6. 0
00	00	00	00	00	00	3C 3C	02AF1	.BYTE	60, 60, 0, 0, 0, 0, 0, 0
						3C 3C 2B 01 00 03	OZAFO D ALIM.	.ASCII	<1>\+\
						0007	02AFB P.AUM: 02AFD	.BYTE	3. 0
00	00	00	00	00	00	3C 3C	02AFF	.BYTE	60. 60. 0. 0. 0. 0. 0. 0
						2D 01 00 03	02807	.ASCII	<1>\-\
						00 03	02B09 P.AUN: 02B0B	.BYTE	3, 0 35 55, 55, 0, 0, 0, 0, 0, 0
00	00	00	00	00	00	37 37	02B0D	BYTE	55, 55, 0, 0, 0, 0, 0, 0
					70	7C 02	02B15 02B18 P.AUO:	.ASCII	<2>\\\\\
						00 03 000F	02B18 P.AUO: 02B1A	.WORD	3, 0
00	00	00	00	00	00	32 32	02B1C	BYTE	50, 50, 0, 0, 0, 0, 0, 0
						3E 01	02B24	.ASCII	<1>\>\
						00 03	02B26 P.AUP: 02B28	.BYTE	3 ₀ 0
00	00	00	00	00	00	32 32	02B2A	BYTE	50, 50, 0, 0, 0, 0, 0, 0
						32 32 3C 01 00 03	02B32	.ASCII	<1>\<\
						00 03	02B34 P.AUQ:	.BYTE	51 0
00	00	00	00	00	00	32 32	02B36 02B38	BYTE	\$1>\<\ 31 50, 50, 0, 0, 0, 0, 0, 0
					00 3E	32 32 5E 02 00 03	02B40	.ASCII	(5)/_>/
						00 03	02B43 P.AUR:	.BYTE	3 ₁ 0
00	00	00	00	00	00	32 32	02B45 02B47	.WORD	50, 50, 0, 0, 0, 0, 0, 0
		••			00 30	32 32 5E 02 00 03 000D	02B4F	.ASCII	<5>/* </td
						00 03	02B52 P.AUS: 02B54	.BYTE	3, 0
						0000	02834	. WUKD	13

DBGPARSER V04-000								J 6 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 Pa 14-Sep-1984 12:17:30 EDEBUG.SRCJDBGPARSER.B32;1	ge 1
		•	00	00 00	00	00	00 32 32 30 01 00 03	02B56	
			00 (00 00	00	00	000E 00 32 32 00 03	02B56	
			00 (00 00	00	00	0015 00 32 32 00 03	02B7E P.AUV: BYTE 3. 0	
			00 (00 00	00	00	0011 00 32 32 00 3E 02 00 03	0288A .ASCII <2>\>=\ 0288D P.AUW: .BYTE 3. 0	
			00 (00 00	00	00	001F 00 2D 2D 26 01 00 03	02891 RVTF 45 45 0 0 0 0 0	
			00 (00 00	00	00	0020 00 28 28 7C 01	02B9F	:
00002A5C 00 00002AB1 00	0002AA3	000029EB 00002A3F 00002A95 00002AEC	000	0029DD 002A31 002A86 002ADD	000 000 000	029CF 02A23 02A78 02ACF	00000019 000029C1 00002A15 00002A6A 00002AC0 00002B18	02BC8 10773, 10787, 10801, 10815, 10830, 10844, - 02BE0 10858, 10872, 10886, 10901, 10915, 10929, - 02BF8 10944, 10959, 10973, 10988, 11003, 11018, -	
			00 (00 00	00	00	01 03 0008 00 00 00 3E 2D 02 00 00	02C14 P.AUY: .BYTE 3, 1 02C16 .WORD 8 02C18 .BYTE 0, 0, 0, 0, 0, 0, 0 02C20 .ASCII <2>\->\ 02C23 P.AVA: .BYTE 0, 0 02C25 .WORD 2	
						00	00 00 00 00 00 00 00 00 00 00 00 29 01 02 00	02C23 P.AVA: BYTE 0, 0 02C25	
						00	00 00 3A 01 00 00 00 00 00 00 2C 01	02C2F .WORD 3 02C31 .BYTE 0, 0, 0, 0 02C35 .ASCII <1>\:\	
						00		02C39 .WORD 1 02C3B .BYTE 0, 0, 0, 0 02C3F .ASCII <1>\ 02C41 .BLKB 3	
			00	002BB4	000	02BAA	00000003 00002BA0 00000000	02C44 .LONG 3 02C48 P.AUZ: .LONG 11168, 11178, 11188 02C54 .LONG 0 02C58 P.AVD: .BLKB 0	
							00000000 01 0E	02C58 .LONG 0 02C5C P.AVE: .BLKB 0 02C5C P.AVF: .BYTE 1 02C5D .BYTE 14	:
							01 005 03 01	02C5D .BYTE 14 02C5E .WORD 5 02C60 .BYTE 3 02C61 .BYTE 1	

	K 6 16-Sep-1984 02:1 14-Sep-1984 12:1	0:13 7:30	VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1	Page 116 (17)
0003	02C62 .WORD 02C64 .BYTE	3		
00 0A 0021	02C65 .BYTE	10		
01	02C66 .WORD 02C68 .BYTE 02C69 .BYTE	1		:
01 0F 0012	OZC6A "WORD	15 18		
00 0A 0021	02C6C .BYTE	10		
0021	02C6E .WORD 02C70 .BYTE	10 33		
01 0E 0005	02C6E .WORD 02C70 .BYTE 02C71 .BYTE 02C72 .WORD	14		
03	UZC/4 .BYIE	3		
0012	02C76 .WORD	18		
02	02C78 .BYTE	1		
000C 06 01	02C7A .WORD 02C7C .BYTE	12		
000C	02C7D .BYTE	12		
07	02C80 .BYTE 02C81 .BYTE 02C82 .WORD	7		
0000	02C82 .WORD 02C84 .BYTE	12		
08 01 000C	02C85 .BYTE	1		
000	02C86 .WORD 02C88 .BYTE	12		
0021	02C89 .BYTE 02C8A .WORD	16		
01	02C8C .BYTE	1		
0000	02C8E .WORD	12		
01	02C91 .BYTE	12		
06	02C94 .BYTE	6		
0000	02C96 .WORD	12		
01	02C98 .BYTE	1.		
000C	02C9A .WORD 02C9C .BYTE	12		
0000	02C9D .BYTE	12		
00	02CAO .BYTE	9		
0021	OZCAZ WORD	33		
OF	OZCAS BYTE	15		
0000 0000 0000 0000 0000 0000 0000 0000 0000	O2CA8 .BYTE	15 18 3 8 3 8 5		
0021	OZCAA .WORD	33		
08	02C8E	5		

DBGPARSER V04-000	M 6	VAX-11 Bliss-32 V4.0-742 Page 118 (17)
00002BC5 00002C61 00002BD9 00002B2D 000029C1 00000001 00000000 00000001 00000000 00002BD9	08 02D30	1. 10689. 11053. 11225, 11361. 11205

V04-000										14-30	:p-17	04 12:17:	. 30	. DEGUG.	SKCJDBO	PARSER.	032,1	
5F000007	23000007	3D042800) :	3E042	800	300	4A800	0	00000000 00000007 2A000000 24000007	02D84 02D88 02D8C P./ 02D84	AVI:	LONG	7 7046430 102368 603979	72, 10 2560, 5	0693811 8720256	2, 1040 7, 1593	459776, - 835527, -	
			00	00	00	00	00 54	00 4F	00 02 0017 08 0B 4E 03 00 03 0018	02DA8 P./ 02DAA 02DAC 02DB4 02DB8 P./	AVK:	BYTE WORD BYTE BYTE WORD BYTE BYTE BYTE WORD BYTE BYTE BYTE WORD BYTE BYTE BYTE BYTE BYTE	23 0	5. 0. 0	. 0. 0.	0, 0		
			00	00	00	00	00	00 4E	0018 0A 0A 41 03 00 03 0019	02DBA 02DBC 02DC4	AVM:	.WORD .BYTE .ASCII .BYTE	24	,°. °.	0.0.	0, 0		
			00	00	00	00	00	00 52	0019 0A 0A 4F 02 02 03 002C 0F 0F 4E 03	02DCA 02DCC 02DD4	AVN:	.WORD .BYTE .ASCII .BYTE	25 10, 10 <2>\OR 3, 2	0. 0.	0. 0.	0.0		
			00	00	00	00	00 54	00 4F		02DD9 02DDB 02DE3 02DE7		.WORD .BYTE .ASCII .BLKB	15, 15 <3>\NO	f\ ^{0, 0,}	0.0.	0.0		
		00002D54	. (00002	D45	000	02035	5	00000004 00002D25 00 03 000E	OZDEC P.	AVJ:	.LONG .LONG .BYTE	11557.	11573,	11589,	11604		
			00	00	00 3D	20	00 54	00 4F	0F 0F 4E 05 00 03 0015	02DFE 02E00 02E08 02E0E P./	AVP:	.BYTE .ASCII .BYTE	15, 15 <5>\NO 3, 0	f =< 0.	0. 0.	0, 0		
			00	00	00 3E	20	00 54	00 4F	0F 0F 4E 05 00 03	02E10 02E12 02E1A 02E20 P./	AVQ:	.ASCII .BYTE	15, 15 <5>\NO 3, 0	f % °.	0. 0.	0.0		
			00	00	00 3C	20	00 54	00 4F	0F 0F 4E 05 00 03 0011 0F 0F 4E 05 01 02 0002 00 00	02E24 02E2C 02E32 P./	AVS:	.BYTE	11	f << 0.	0. 0.	0.0		
			00	00	00	00	00	00	00 00 5C 01 01 03	02E22 02E24 02E2C 02E32 02E34 02E36 02E36 02E40 P./	AVT:	.BYTE	51><92: 3, 1	0. 0. 0	. 0. 0.	0		
			00	00	00	00	00	00	5C 01 01 03 0003 00 00 5C 01 01 04 0004 00 00 28 01 02 02	02E44 02E4C	AVU:	.BYTE .ASCII .BYTE	3	0. 0. 0	. 0. 0.	0		
			00	00	00	00	00	00	0004 00 00 28 01 02 02	02E50 02E52 02E5A 02E5C P.	AVV:	.WORD .BYTE .ASCII .BYTE	4	0. 0. 0	. 0. 0.	0		
			00	00	00	00	00	00	0035	02E5E 02E60 02E68	AVW:	.WORD .BYTE .ASCII .BYTE	63 15, -5(<1>\>\	5. 0. 0	. 0. 0.	0.0		
			00	00	00	00	00	00	0040	02E6C 02E6E 02E76	AVX:	WORD BYTE	<1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	5. 0. 0	. 0. 0.	0.0		

DBGPARSER V04-000 B 7 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 120 (17)
00 00 00 00 00 00 00 00 00 00 00 00 00	
02 02 02E86 P.AVY: BYTE 2, 2 000B 02E88WORD 11 00 00 00 00 00 00 08 05 02E8ABYTE 5, -56, 0, 0, 0, 0, 0 28 01 02E92ASCII <1>\(\)	
00 00 00 00 00 00 06 C8 02E98 .WORD 12 .WORD 12 .BYTE -56, 6, 0, 0, 0, 0, 0, 0 .ASCII <1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	
00 00 00 00 00 1E 1E 02EA6 .WORD 9 2F 01 02EAE .ASCII <1>\/\ 00 02 02EBO P.AWB: .BYTE 2, 0 0004 02EB2 .WORD 4	
00 00 00 00 00 00 08 19 02EB2 .WORD 4 .BYTE 25, -56, 0, 0, 0, 0, 0, 0 2B 01 02EBC .ASCII <1>\+\ 00 02 02EBE P.AWC: .BYTE 2, 0	
00 00 00 00 00 00 08 19 02EC2 .BYTE 25, -56, 0, 0, 0, 0, 0, 0 0 2D 01 02ECA .ASCII <1>\-\	
00 00 00 00 00 00 14 14 02ED0 .BYTE 20, 20, 0, 0, 0, 0, 0 0 2B 01 02ED8 .ASCII <1>\tag{1} \tag{1} \tag	
00 00 00 00 00 00 14 14 02EDC .WORD 7 .BYTE 20, 20, 0, 0, 0, 0, 0, 0 2D 01 02EE6 .ASCII <1>\-\ 00 03 02EE8 P.AWF: .BYTE 3, 0 .WORD 15	
00 00 00 00 00 00 0F OF OZEEC .BYTE 15, 15, 0, 0, 0, 0, 0	
00 00 00 00 00 00 0F OF OZEFA .BYTE 15, 15, 0, 0, 0, 0, 0, 0 3C 01 02F02 .ASCII <1>\<\\\ 00 03 02F04 P.AWH: .BYTE 3, 0 000D 02F06 .WORD 13	
00 00 00 00 00 00 0F 0F 0F 02F08 .BYTE 15, 15, 0, 0, 0, 0, 0, 0 3D 01 02F10 .ASCII <1>\=\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	
00002E49 00002E3B 00002E2D 00002E1F 00002E11 00002E03 02F30 11779, 11793, 11807, 11821, 11835, 1184 00002E81 00002E73 00002E65 00002E57 02F48 11863, 11877, 11891, 11905 00 03 02F58 P.AWI: BYTE 3, 0	9; -
00 00 00 00 00 1E 1E 02F5C BYTE 30, 30, 0, 0, 0, 0, 0	
00 00 02F66 P.AWK: .BYTE 0, 0 0002 02F68 .WORD 2 00 00 00 02F6A .BYTE 0, 0, 0, 0 29 01 02F6E .ASCII <1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	

DB VO

DBGPARSER V04-000	C 7 16-Sep-1984 02: 14-Sep-1984 12:	10:13 VAX-11 Bliss-32 V4.0-742 17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 121 (17)
00 0	74 01 02570 4551	I \$1>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	
00 0	0001 02F7C .WORD 0 00 00 02F7E .BYTE 2C 01 02F82 .ASCI	1 51>0.0.0	
00002EF7 00002EED	00 00 02F7A P.AWM: BYTE 0001 02F7C	12003, 12013, 12023	•
	00000000 02F98 .LONG 02F9C P.AWO: .BLKB 02 02F9C P.AWP: .BYTE 01 02F9D .BYTE	Ŏ Q	:
	01 02F9D .BYTE 0006 02F9E .WORD 03 02FAO .BYTE	1 6 3	
	0006 02F9E .WORD 03 02FAO .BYTE 03 02FA1 .BYTE 0008 02FA2 .WORD 09 02FA4 .BYTE 0F 02FA5 .BYTE 0008 02FA6 .WORD 04 02FA8 .BYTE	8 9 15	
	0006 02F9E .WORD 03 02FA0 .BYTE 03 02FA1 .BYTE 0008 02FA2 .WORD 09 02FA4 .BYTE 0F 02FA5 .BYTE 0008 02FA6 .WORD 04 02FA8 .BYTE 07 02FA9 .BYTE	8 7 13	
	01 02FAC .BYTE 0C 02FAD .BYTE 0010 02FAE .WORD 00000000 02FBO .LONG	12 16	
	01 02FAC .BYTE 00 02FAD .BYTE 0010 02FAE .WORD 00000000 02FB0 .LONG 01 02FB4 .BYTE 02 02FB5 .BYTE 0010 02FB6 .WORD 00000000 02FB8 .LONG 03 02FBC .BYTE 0E 02FBD .BYTE 0008 02FBE .WORD 09 02FC0 .BYTE	0 1 2 16	
	0010 02FB6 .WORD 00000000 02FB8 .LONG 03 02FBC .BYTE 0E 02FBD .BYTE	0 3 14 8	
	03 02FBC .BYTE 0E 02FBD .BYTE 0008 02FBE .WORD 09 02FC0 .BYTE 0F 02FC1 .BYTE	8 15	
	0008 02FC2 .WORD 04 02FC4 .BYTE 12 02FC5 .BYTE 000D 02FC6 .WORD	2 18 13	
	01 02FC8 .BYTE 17 02FC9 .BYTE 0010 02FCA .WORD 00000000 02FCC .LONG	1 23 16	
	04 02FD0 .BYTE 26 02FD1 .BYTE 000D 02FD2 .WORD	38 13	
	0000 02FD2 .WORD 01 02FD4 .BYTE 2A 02FD5 .BYTE 0010 02FD6 .WORD	12	
00002F05 00002F19 00002BD9 00002E95 00002D69 00000000 00000000 00000001 00000000 000028D1	0010 02FD6 .WORD 00000000 02FD8 .LONG 00000000 02FDC .LONG 00002D09 02FE0 P.AWQ: .LONG 00002F15 02FF8 00000000 03010	11529, 11625, 11925, 11225, 12057, 120 12053, 10449, 0, 1, 0, 0, 0	37

DE

```
DBGPARSER
                                                                        16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                  VAX-11 Bliss-32 V4.0-742 [DEBUG.SRCJDBGPARSER.B32;1
                                                                                                                                           Page 122
(17)
V04-000
                                                                  03014 LANGUAGE_TABLE_PTRS:

1000 9709, 8945, 6393, 8137, 5437, 11473, -

10585, 7237, 12125, 4649, 3489
00002CD1
           0000153D
                      00001FC9 000018F9
                                            000022F1
                                                        000025ED
           00000DA1
                      00001229
                                                        00002959
                                 00002F5D
                                             00001C45
                                                                                   .PSECT DBG$OWN, NOEXE, PIC, 2
                                                                   00000 ADDRESS_LENGTH:
                                                                                    BLKB
                                                                   00004 ADDRESS_TYPE:
                                                                                    BLKB
                                                                   00008 BIF_TABLE:
                                                                   OOOOC CASING_SIGNIFICANT:
                                                                                   .BLKB
                                                                   00010 CHARPTR: BLKB
                                                                   00014 CHARTBL: BLKB
                                                                                            1024
                                                                   00414 COMPONENTS IN PATHNAME :
                                                                                    BEKB
                                                                   00418 ENFORCE_RECORD:
                                                                                   BLKB
                                                                   0041C EXPRESSION RADIX:
                                                                                    BEKB
                                                                   00420 IDENT_OPERATOR_TABLE:
                                                                                   BLKB
                                                                   00424 INCOMPLETE_QUAL:
                                                                                   .BEKB
                                                                   00428 MULTIPLE_SUBSCR:
                                                                   0042C OPCHAR_OPERATOR_TABLE:
                                                                   00430 PRIMARY TABLE:
                                                                                   .BLKB
                                                        00000000
                                                                   00434 SAVED_TOKEN:
                                                                   00438 STATE_TABLE:
                                                                                   .BLKB
                                                                   0043C SUBSCRIPT TERM_TBL:
                                                                   00440 PRIDTBL: BLKB
                                                        00000000
                                                                   00444 TERMINATOR_CODE:
                                                                                    LONG
                                                                   00448 TERMINATOR_LENGTH:
                                                        00000000
                                                                                    LONG
                                                                   0044C VARSTACK
                                                                                            80
                                                                   0049C VARSTACK2:
                                                                                            80
                                                                   004EC VARSTACK3:
                                                                                   BLKB
                                                                                            80
                                                                   0053C VARSTK_INDEX:
                                                                                   .BLKB
                                                                                   .PSECT
                                                                                           DBG$GLOBAL, NOEXE, PIC, 2
                                                                   00000 DBG$GL_CHARTBL::
```

.BLKB

DI

V

P.AGQ

Page 123 (17)

DE

D

Page 124 (17)

Page 125 (17)

```
H 7
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                           VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1
          RPG_IDENT_OPTBL= P.AVJ
RPG_NOT_EQL_TOKEN= P.AVO
RPG_NOT_GTR_TOKEN= P.AVP
RPG_NOT_LSS_TOKEN= P.AVQ
RPG_OPCHAR_OPTBL= P.AVR
RPG_MULTIPEY_TOKEN= P.AWI
RPG_SUBSCR_TERM_TBL=P.AWJ
RPG_PRID_TABLE= P.AWN
RPG_PRID_TABLE= P.AWO
RPG_NUMBER_TABLE= P.AVF
RPG_PRIMARY_TABLE= P.AWP
RPG_TABLES= P.AWQ
.EXTRN_DBG$DATA_L
                                                                                                                          LE = P.AWP
P.AWQ
DBG$DATA_LENGTH
DBG$DEF_SYM_FIND
DBG$DUMP_HER, DBG$ENUM_POS
DBG$ENUM_VAL, DBG$EVALDP_SET_LANGUAGE
DBG$EVAL_ADA_TICK
DBG$EVAL_ADA_TICK
DBG$EVAL_ANG_OPERATOR
DBG$EVAL_ANG_OPERATOR
DBG$EVAL_ANG_OPERATOR
DBG$GET_TEMPMEM
DBG$MASR_FIND, DBG$HASH_FIND_SETUP
DBG$MAKE_SKELETON_DESC
DBG$MAP_DTYPE_CLASS
DBG$MAP_DTYPE_CLASS
DBG$NCOPY_DESC, DBG$REWLINE
DBG$NPATHDESC_TO_CS
DBG$NOM_BYTES, DBG$PRIM_TO_ADDR
DBG$PRINT, DBG$PRINT_SET_LANGUAGE
DBG$STA_SYMSIZE
DBG$STA_SYMSIZE
DBG$STA_SYMSIZE
DBG$STA_SYMSIZE
DBG$STA_SYMSIZE
DBG$STA_TYP_FCODE
DBG$STA_TYP_FCODE
DBG$STA_TYP_FCODE
DBG$STA_TYP_FILE
DBG$STA_TYP_TECORD
DBG$STA_TYP_TECORD
DBG$STA_TYP_TECORD
DBG$STA_TYP_TECORD
DBG$STA_TYP_TECORD
DBG$STA_TYP_SUBRNG
DBG$STA_TYP_SUBRNG
DBG$STA_TYP_VARIANT
DBG$STA_TYP_VARIANT
DBG$STA_TYP_VARIANT
DBG$STA_TYP_VARIANT
DBG$STA_TYP_VARIANT
DBG$STA_TYP_VARIANT
DBG$STA_TYP_VARIANT
DBG$STA_TYP_VARIANT
DBG$STA_TYP_VARIANT
DBG$STA_TYP_TYPEDPTR
DBG$STA_TYP_VARIANT
DBG$STA_TYP_VARIANT
DBG$STA_TYP_VARIANT
DBG$STA_TYP_TYPEDPTR
DBG$STA_TYP_VARIANT
DBG$STA_TYP_TYPEDPTR
DBG$STA_TYP_VARIANT
DBG$STA_TYP_TYPEDPTR
DBG$STA_
                                                                        .EXTRN
                                                                         .EXTRN
                                                                          .EXTRN
                                                                        .EXTRN
                                                                          .EXTRN
                                                                         .EXTRN
                                                                         .EXTRN
                                                                         .EXTRN
                                                                         .EXTRN
                                                                         .EXTRN
                                                                         .EXTRN
                                                                         .EXTRN
                                                                          .EXTRN
                                                                          .EXTRN
                                                                          .EXTRN
                                                                         .EXTRN
                                                                         .EXTRN
                                                                         .EXTRN
                                                                         .EXTRN
                                                                          .EXTRN
                                                                         .EXTRN
                                                                         .EXTRN
                                                                         .EXTRN
                                                                         .EXTRN
                                                                          .EXTRN
                                                                          .EXTRN
                                                                          .EXTRN
                                                                          .EXTRN
                                                                          .EXTRN
                                                                          .EXTRN
                                                                         .EXTRN
                                                                          .EXTRN
                                                                           .EXTRN
                                                                          .EXTRN
                                                                          .EXTRN
                                                                          .EXTRN
                                                                        .EXTRN
.EXTRN
.EXTRN
                                                                       .EXTRN
                                                                         .EXTRN
```

Page 126 (17) DBGPARSER V04-000

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

0000 00000 AAA_DUMMY: WORD Save nothing

; Routine Size: 3 bytes, Routine Base: DBG\$CODE + 0000

GLOBAL ROUTINE DBG\$ADDR_EXP_INT(INPUT_DESC, ADDR_EXP_PTR, TYPE, LENGTH, RADIX, TERM_INDEX) =

This is the Address Expression Interpreter for most languages supported by DEBUG. It parses and evaluates a DEBUG address expression and returns an Address Expression Descriptor which represents the value of the expression. This routine itself is only a set-up routine which sets up the character pointer and the expression radix to use and then calls DBG\$EXPRESSION_PARSER to do the actual work.

INPUTS

- INPUT_DESC A string descriptor which points to the input string to be parsed as an Address Expression. Only the pointer field of this descriptor is actually used--the string is expected to terminated by a carriage-return character.
- ADDR_EXP_PTR The address of a longword location to receive a pointer to a Primary or Value Descriptor returned by this routine.
- TYPE The address of a longword location to receive the address "type", namely instruction or not instruction.
- LENGTH The address of a longword location to receive the length of the current instruction if the "type" is instruction.
- RADIX The radix to be used to interpret integer numbers. The allowed radix values are DBG\$K_DECIMAL, DBG\$K_HEX, DBG\$K_OCTAL, and DBG\$K_BINARY.
- TERM_INDEX A "terminator index" which indicates which lexical tokens are allowed as expression terminators in this context. For example, in the EXAMINE command, "," and ":" are allowed terminators and in the DEPOSIT command, "=" is the allowed terminator. These index values have names of the form TOKEN\$K_TERM_xxx.

OUTPUTS

- INPUT_DESC The input string descriptor is updated to point to the first character after the address expression just parsed. If the parse was stopped by a terminator token, the input string descriptor will point to that token on return.
- ADDR_EXP_PTR A Primary or Value Descriptor is constructed and a pointer to that descriptor is returned to ADDR_EXP_PTR.
- TYPE If the address expression yields an instruction address, the value DBG\$K_INSTRUCTION is returned to TYPE. Otherwise, the value DBG\$K_NOTYPE is returned to TYPE.
- LENGTH If the value DBG\$K_INSTRUCTION is returned to TYPE, the length in bytes of the instruction pointed to by the address expression is returned to LENGTH. Otherwise, 0 is returned.
- The value STS\$K_SUCCESS is return as the routine result if the expression was terminated by a carriage-return character. If it

RETURN STS\$K_SUCCESS;

000C 00000 9E 00002 D0 00009 D0 0000D

END:

```
DBG$ADDR_EXP_INT, Save R2,R3
CHARPTR, R3
INPUT_DESC, R2
4(R2), CHARPTR
ENTRY
MOVAB
MOVL
MOVL
```

VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1

3473

Page 129 (18)

DBGPARSER V04-000					L 7 16-Sep 14-Sep	-1984 02:10:1 -1984 12:17:3	VAX-11 Bliss-32 V4.	0-742 Page 130 R.B32;1 (18)
		040C	C3	14 AC	DO 00011	MOVL R	RADIX, EXPRESSION_RADIX	: 3474
		F4	A3	0424 C3 80 8E	9A 0001B	MOVL R CLRL S MOVZBL A	1128, ADDRESS_TYPE	348
			50	F0 A3 18 AC 00000000'EF40	04 00020 00 00023 00 00027 9F 0002F	CLRL AMOVL TO MOVE TO PUSHAB TO PUSHL AMOVE REMOVE REMOVE REMOVE REMOVE REMOVE AMOVE AMOVE AMOVE REMOVE REM	SAVED_TOKEN \$128, ADDRESS TYPE ADDRESS LENGTH FERM_INDEX, RO FERM_POINTER_TBL[RO], RO FABLEBASE[RO]	3474 3475 3484 3484 3493
			50	00000000 EF 40	9F 0002F	PUSHAB T	TABLEBASE[RO]	
		0000V	CF	01 02	DD 00036 FB 00038	CALLS A	2. DBGSEXPRESSION PARSER	3492
		08	BC 51	50	DO 0003D 3C 00041	MOVL R MOVZWL (1031 01	3499
			51	04 A2	CO 00044 DO 00048 A3 0004B	ADDL2 4	(R2), R1 CHARPTR, R0 R0, R1, (R2) R0, 4(R2) ADDRESS_TYPE, aTYPE ADDRESS_LENGTH, aLENGTH (R0), #T3	
	62	04	51 A2	50 50	DO 0004F	MOVL C SUBW3 R MOVL R	RO, R1, (R2) RO, 4(R2)	3496
		04 00 10	BC BC OD	F4 A3 F0 A3	00 00053 00 00058 91 00050	MOVL A MOVL A CMPB (BNEQ 1	ADDRESS TYPE, aTYPE	3496 3498 3498
			ŌĎ	FO A3 60 04	91 0005D 12 00060	CMPB ((RO), #T3	3499
			50	ŎĨ	00 00062 04 00065	MOVL A	1\$ V1, RO	3500
				50	04 00066 1\$: 04 00068	CLRL R	80	3502

; Routine Size: 105 bytes, Routine Base: DBG\$CODE + 0003

Page 131

GLOBAL ROUTINE DBG\$BUILD_PRIMARY_SUBNODE(PRIMPTR, KIND, SYMID, FCODE, TYPEID, DSTPTR): NOVALUE =

This routine constructs a Primary Descriptor Sub-Node for a specified symbol and appends this sub-node to a specified Primary Descriptor. If the symbol is an array or a record, an Array or Record Sub-Node is constructed and the specific information needed for those data types is filled in. Otherwise, a Normal Sub-Node is constructed.

INPUTS

PRIMPTR - A pointer to the Root Node of the Primary Descriptor to which the Sub-Node should be appended.

KIND - The KIND of the symbol for which the Primary Descriptor Sub-Node should be constructed. This is the RST "kind" returned by DBG\$STA_GETSYMBOL.

SYMID - The SYMID of the symbol for which the Primary Descriptor Sub-Node should be constructed. If there is no SYMID (as for an individual array element, for exmple), SYMID should be zero.

FCODE - The FCODE ("format code") for the data type of the symbol for which the Primary Descriptor Sub-Node should be constructed. If the symbol is not a data items (i.e., if its KIND is not RST\$K_DATA or RST\$K_TYPCOMP), FCODE should be zero.

TYPEID - The Type ID of the data item for which the Primary Descriptor Sub-Node should be constructed. If the entity in question is not a data item, TYPEID should be zero.

DSTPTR - A pointer to the DST record corresponding the the data item. This is used in the case of BLISS data to obtain the information about what kind of BLISS structure this is.

OUTPUTS

A Primary Descriptor Sub-Node is created and appended to the PRIMPTR Primary Descriptor. PRIMPTR itself is not changed, however. There is no other output.

BEGIN

MAP

DSTPTR: REF DST\$RECORD, PRIMPTR: REF DBG\$PRIMARY, SYMID: REF RST\$ENTRY; ! Pointer to DST record ! Pointer to Primary Descriptor ! Pointer to symbol's RST entry

BUILTIN INSQUE:

! Insert-Queue function

MAX_DIMS = 20;

! Maximum dimension count we allow

ATOMIC_TYPE,

! A dtype code

Array element bit length Array or record component bit-size (not actually used) Pointer to bounds vector in array descriptor TYPEID for array element cell type Record Component vector (not used) Record Component vector (not used)
Pointer to vector of subscript TYPEIDs
Pointer to array descriptor
Higher bound of the subrange value
Byte length of array element
Lower bound of the subrange value
Number of record components (not used)
Number of array dimensions
Pointer to created Primary Sub-Node
Computed offset for ARRAY[0,0,...,0]
Size in bits of the data type
Vector of strides computed from
multipliers in array descriptor
Pointer to stride or multiplier
vector in array descriptor vector in array descriptor Size of stride for BLISS vectors Pointer to TYPEID RST entry Pointer to subscript block-vector in Primary Descr Array Sub-Node

Page 132 (19)

Determine what kind of symbol this is and allocate the Sub-Node memory block accordingly. If this symbol is an array or a record, we also must collect the special information needed for those data types and fill it

Handle arrays. Determine the number of dimensions in the array and allocate an Array Sub-Node large enough to hold that many dimensions. Then fill in all the fields specific to the Array Sub-Node.

Get all needed type information about the array type from DBG\$STA_TYP_ARRAY. Allocate a memory block for the Array Sub-Node and fill in the fields in the fixed part of that Sub-Node.

DBG\$STA_TYP_ARRAY(.TYPEID, DSCADDR, CELLTYPE NDIMS, DIMVECPTR, BITSIZE);

```
DBGPARSER
V04-000
                                                                               23456789012345678901234567890123456789012345678901234567890123456789012345678
```

```
If we got a symid passed in to this routine, and the symid represents an array, try calling SYMVALUE to get an array
   descriptor for this array. If we get one, then use this descriptor instead of the one we got back from STA_TYP_ARRAY.
   Note - normally, these 2 descriptors will be the same.
However, for dynamic arrays in PASCAL, the runtime descriptor
   (which we get back when we call SYMVALUE with the symid) is
   correct, but the compile-time descriptor (which is part of
   the typespec) is wrong. This code is a workaround for this problem in the PASCAL DST.
    .SYMID NEQ O
THEN
      BEGIN
      LOCAL
            DESC: VECTOR[3],
RSTPTR: REF RSTSENTRY,
     VALUE_KIND;

RSTPTR = .SYMID;

WHILE .RSTPTR[RST$B_KIND] NEQ RST$K_MODULE DO

RSTPTR = .RSTPTR[RST$L_UPSCOPEPTR];

IF .RSTPTR[RST$B_LANGUAGE] EQL DBG$K_PASCAL
      THEN
            BEGIN
            DBG$STA_SETCONTEXT(.SYMID);
DBG$STA_SYMVALUE(.SYMID, DESC, VALUE_KIND);
IF .VALUE_KIND EQL DBG$K_VAL_DESCR
                  DSCADDR = .DESC[0]:
            END:
      END:
IF .NDIMS GTR MAX_DIMS THEN SIGNAL (DBG$ TOOMANDIM);
NODEPTR = DBG$GET_TEMPMEM(DBG$K_PRIM_SIZE_ARRAY +
                                                        .NDIM5*DBG$K_PRIM_SIZE_SUBS);
NODEPTR[DBG$B_PNARR_DIMCNT] = .NDIMS;
!*** The following is a temporary workaround to a problem in
!*** the PASCAL DST: They are giving us array descriptors with
!*** class UBA but dtype=0. Since dtype must be VU for this
!*** class, we fill in the correct dtype here in this case.
IF .DSCADDR[DSC$B_CLASS] EQL DSC$K_CLASS_UBA
      NODEPTR[DBG$B_PNARR_DTYPE] = DSC$K_DTYPE_VU
ELSE !*** End temporary workaround.
      NODEPTR[DBG$B_PNARR_DTYPE] = .DSCADDR[DSC$B_DTYPE];
NODEPTR[DBG$W_PNARR_LENGTH] = .DSCADDR[DSC$W_LENGTH];
NODEPTR[DBG$B_PNARR_SCALE] = .DSCADDR[DSC$B_SCALE];
```

```
C 8
16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1
```

NODEPTR[DBG\$B_PNARR_DIGITS] = .DSCADDR[DSC\$B_DIGITS]; NODEPTR[DBG\$V_PNARR_COLUMN] = .DSCADDR[DSC\$V_FL_COLUMN]; NODEPTR[DBG\$L_PNARR_CELLTYPE] = .CELLTYPE;

Set up pointers to the stride (or multiplier) vector and to the bounds vector in the array descriptor.

STRIDEPTR = .DSCADDR + 20; BOUNDVEC = .STRIDEPTR + 4*.NDIMS;

Determine what kind of array descriptor we have. Determine whether we have multipliers or strides and whether we are doing byte or bit addressing.

CASE _DSCADDR[DSC\$B_CLASS] FROM DSC\$K_CLASS_S TO DSC\$K_CLASS_UBA OF

Handle an ordinary Array Descriptor. This descriptor has byte multipliers. Since we only allow strides in the Array Sub-Node, we convert the array multipliers into strides here. (Note that STRIDEPTR points to multipliers in this case.)

COMPUTE_STRIDES: BEGIN

!*** Temporary workaround to a problem in the PL/I DST.
!*** The multipliers that we are getting in the array
!*** descriptors are not correct. They are giving us
!*** strides instead of multipliers. So we skip the
!*** computation of strides if language is PL/I.

IMP_SYMID = .PRIMPTR[DBG\$L_DHDR_SYMID0];

TMP_SYMID = .PRIMPTR[DBG\$L_DHDR_SYMIDO];
IF .TMP_SYMID NEQ 0
THEN

BEGIN
WHILE .TMP SYMID[RST\$B KIND] NEQ RST\$K MODULE DO

TMP SYMID = .TMP SYMID[RST\$L UPSCOPEPTR];

IF (.TMP_SYMID[RST\$B_LANGUAGE] EQL DBG\$K_PLI) AND

(.TMP_SYMID[RST\$V_OLDPLIFLAG])

THEN

LEAVE COMPUTE_STRIDES;

! Pick up the array element length in bytes. We need this length to compute strides properly below.

LENGTH = DBG\$DATA_LENGTH(.DSCADDR); LENGTH = (.LENGTH + 7)/8;

! If this is a column-major order array, we compute the ! array's stride values from its multiplier values.

```
DE
```

```
DBGPARSER
V04-000
                                                                             16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                          VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1
  IF .DSCADDR[DSC$V_FL_COLUMN]
                                                         BEGIN

STRIDE[0] = .LENGTH;

INCR I FROM 1 TO .NDIMS - 1 DO

STRIDE[.I] = .STRIDE[.I - 1]*.STRIDEPTR[.I - 1];
                                                         END
                                                       If this is a row-major order array, we compute the
                                                       strides in the opposite direction.
                                                    ELSE
                                                         BEGIN

STRIDE[.NDIMS - 1] = .LENGTH;

DECR I FROM .NDIMS - 2 TO 0 DO

STRIDE[.I] = .STRIDE[.I + 1]*.STRIDEPTR[.I + 1];
                                                         END:
                                                       Make STRIDEPTR point to the newly computed stride vector.
                                                     STRIDEPTR = STRIDE[0]:
                                                     END:
                                                  Handle a Noncontiguous Array Descriptor or a Varying String
                                                  Descriptor. Here we already have byte strides instead of
                                                  multipliers so there is nothing we need to do.
                                                DSCSK_CLASS_NCA,
DSCSK_CLASS_VSAJ:
                                                  Handle an Unaligned Bit Array Descriptor. This descriptor
                                                  is like the Noncontiguous Array Descriptor except that it has
                                                  bit strides instead of byte strides.
                                                [DSC$K_CLASS_UBA]:
                                                    NODEPTREDBG$V_PNARR_BITREF] = TRUE;
                                                  Any other case constitutes an invalid array descriptor.
                                                  Signal an error message.
                                                [INRANGE, OUTRANGE]:
                                                     SIGNAL (DBG$_INVARRDSC);
                                                TES:
                                             Loop through the dimensions of the array to set up the subscript
```

block-vector in the Array Sub-Node. In that vector, we set the

```
DE
```

Page 136 (19)

```
E 8
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                               VAX-11 Bliss-32 V4.0-742
LDEBUG.SRCJDBGPARSER.B32;1
                          subscript value to be the same as the lower bound--this is changed later when the actual subscript value is picked up. The lower and
  upper bounds, the stride, and the subscript type TYPEID are all picked up and stored away as well.
                                                          SUBVECTOR = NODEPTR[DBG$A_PNARR_SVECTOR];
OFFSET = 0;
INCR I FROM 0 TO .NDIMS - 1 DO
                                                                 BEGIN
                                                                 SUBVECTOR[.1, DBG$L_PNSUB_SVALUE] = .BOUNDVEC[2*.1];
SUBVECTOR[.1, DBG$L_PNSUB_STRIDE] = .STRIDEPTR[.1];
SUBVECTOR[.1, DBG$L_PNSUB_LBOUND] = .BOUNDVEC[2*.1];
SUBVECTOR[.1, DBG$L_PNSUB_UBOUND] = .BOUNDVEC[2*.1] + 1];
                                                                SUB_TYPEID = 0:
IF .DIMVECPTR[.I] NEQ 0
THEN
                                                                       BEGIN
                                                                       SUB_TYPEID = .DIMVECPTR[.I];
WHILE .SUB_TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG DO
DBG$STA_TYP_SUBRNG(.SUB_TYPEID,SUB_TYPEID,LOWPTR,HIGHPTR,SIZE);
                                                                       SUBVECTOR[.I, DBG$L_PNSUB_TYPEID] = .SUB_TYPEID; END
                                                                ELSE
                                                                       SUBVECTOR[.I, DBG$L_PNSUB_TYPEID] = .DIMVECPTR[.I];
                                                                OFFSET = .OFFSET - (.SUBVECTOR[.I, DBG$L_PNSUB_STRIDE]*
.BOUNDVEC[2*.1]);
                                                                   One additional thing needs to be done for arrays indexed by enumeration types in ADA. The SVALUE field needs
                                                                    to be filled in to be the value of the enumeration,
                                                                   not its position.
                                                                     .DBG$GB_LANGUAGE EQL DBG$K_ADA
                                                                 THEN
                                                                            .SUB_TYPEID NEQ 0
                                                                       THEN
                                                                                   .SUB_TYPEID[RST$B_FCODE] EQL RST$K_TYPE_ENUM
                                                                              THEN
                                                                                    SUBVECTOR[.I, DBG$L PNSUB_SVALUE] = DBG$ENUM_VAL(.SOB_TYPEID, .SUBVECTOR[.I, DBG$L_PNSUB_SVALUE]);
                                                                 END:
                                                             Finally fill in the offset from the start of the array to
                                                             element ARRAY[0,0,...,0]. Also mark symbol as an aggregate.
                                                          NODEPTR[DBG$L_PNARR_OFFSET] = .OFFSET;
IF .DBG$GL_ARRSUB_F[AG
THEN
```

PRIMPTR[DBG\$V_DHDR_AGGR] = TRUE;

IF .DSTPTR [DST\$V_BLI_REF] AND .SYMID NEQ O

Page 137 (19)

VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32:1

```
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
      each element as a longword. If it turns out
      we are not doing aggregate examine, then we fix up this information in the GET_BLISS_SUBSCRIPTS
      routine.
  SUBVECTOR [0, DBG$L PNSUB STRIDE] = 4;
NODEPTR [DBG$B PNARR DTYPE] = DSC$K_DTYPE_L;
NODEPTR [DBG$W_PNARR_LENGTH] = 4;
      The upper bound on subscripts is one less than the number of units that were allocated in the declaration of the vector. (Origin-O subscripting) We temporarily dummy this up as if stride were 4, for
      purposes of aggregate output.
   STRIDE_SIZE = .DSTPTR [DST$V_BLI_BLOCK_UNIT_SIZE];
IF (.DSTPTR [DST$L_BLI_BLOCK_UNITS] EQ[ 0) OR
(.STRIDE_SIZE EQL 0)
         SUBVECTOR [O, DBG$L_PNSUB_UBOUND] = 0
   ELSE
         SUBVECTOR [O, DBG$L_PNSUB_UBOUND] =
             (.DSTPTR [DST$L_BEI_BLOCK_UNITS] *. STRIDE_SIZE-1)
   END:
Handle blockvectors. We allocate enough space for an
Array Sub-Node with one subscript. The fields of the
subnode are then filled in.
```

CDST\$K_BLI_BLKVEC]:

BEGIN

PRIMPTR[DBG\$V_DHDR_BLIBLK] = TRUE;

FCODE = RST\$K_TYPE_ARRAY;

NODEPTR = DBG\$GET_TEMPMEM (DBG\$K_PRIM_SIZE_ARRAY +

2 * DBG\$K_PRIM_SIZE_SUB\$);

NODEPTR [DBG\$B_PNARR_DIM(NT] = 2;

NODEPTR [DBG\$B_PNARR_DTYPE] = D\$C\$K_DTYPE_V;

NODEPTR [DBG\$W_PNARR_LENGTH] = 32;

SUBVECTOR = NODEPTR [DBG\$A_PNARR_SVECTOR];

The stride on the first subscript is the stride on the second subscript times the number of units per block. The upper bound on the first subscript depends on the number of blocks in the blockvector.

SUBVECTOR [O, DBG\$L PNSUB STRIDE] =

.DSTPTR [DST\$B_BLI_BLKVEC_UNIT_SIZE] *

.DSTPTR [DST\$L_BLI_BLKVEC_BLOCKS] EQL 0 THEN SUBVECTOR [O, DBG\$L_PNSUB_UBOUND] = 0

END:

```
DBGPARSER
V04-000
                                                                                           16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                              VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32;1
  4073
4074
4075
4076
4077
4078
4081
4082
4083
                                                   IF .FCODE EQL RSTSK_TYPE_PTR THEN
                                                         BEGIN
                                                         ATOMIC_TYPE = DSC$K_DTYPE_L;
BIT_LENGTH = 32;
TYPEID = DBG$TYPEID_FOR_ATOMIC(.ATOMIC_TYPE, .BIT_LENGTH, FALSE);
                                                         END:
                                                   END:
                                                for all other cases, just allocate a Normal Sub-Node. Also
  4084
                                                clear the aggregate flag.
  4085
  4086
                                             [INRANGE, OUTRANGE]:
  4087
  4088
                                                   NODEPTR = DBG$GET_TEMPMEM(DBG$K_PRIM_SIZE_NORMAL);
PRIMPTR[DBG$V_DHDR_AGGR] = FALSE;
  4089
  4090
                                                   END:
  4091
  4092
                                             TES:
  4094
  4095
                                        ! Fill in the standard fields common to all Primary Descriptor Sub-Nodes.
  4096
                                       NODEPTR[DBG$B_PNODE_KIND] = .KIND;

NODEPTR[DBG$B_PNODE_FCODE] = .FCODE;

NODEPTR[DBG$L_PNODE_TYPEID] = .TYPEID;

NODEPTR[DBG$L_PNODE_SYMID] = .SYMID;

NODEPTR[DBG$L_PNODE_RELOC] = 0;
  4097
  4098
 Also set the final Sub-Node's KIND, FCODE, and TYPEID in the Primary
                                          Descriptor Root Node. The Root Node thus describes the object described
                                          by the Primary Symbol as a whole.
                                       PRIMPTR[DBG$B_DHDR_KIND] = .KIND;
PRIMPTR[DBG$B_DHDR_FCODE] = .FCODE;
PRIMPTR[DBG$L_DHDR_TYPEID] = .TYPEID;
                                          Append the Sub-Node to the Primary Descriptor by linking it in at the
                                          end of the doubly linked Sub-Node chain. Then return.
                                        INSQUE(.NODEPTR,.PRIMPTR[DBG$L_PRIM_BLINK]);
                                        RETURN;
                                        END:
                                                                                                                     DBG$BUILD_PRIMARY_SUBNODE, Save R2,R3,R4,-
                                                                              OFFC 00000
                                                                                                          .ENTRY
                                                                                                                                                                                    : 3503
                                                                                                                     R5,R6,R7,R8,R9,R10,R11
-140(SP), SP
PRIMPTR, R8
                                                                                     00002
                                                                                                          MOVAB
                                                                                                                                                                                      3591
                                                                                                         MOVL
```

DBGPARSER V04-000		M 8 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 144 (19)
002C 002C 002C 002C	00000000G 00 01 002C 002C 002C 002C 002C 002C 002C 002C	10	3598
	00000000G 00 55 04 A8	2\$-1\$,- 2\$-1\$ 06 DD 00043 2\$: PUSHL #6 01 FB 00045 CALLS #1, DBG\$GET_TEMPMEM 50 DO 0004C MOVL RO, NODEPTR 01 8A 0004F BICB2 #1, 4(R8) 047B 31 00053 BRW 66\$ 24 AE 9F 00056 3\$: PUSHAB BITSIZE 04 AE 9F 00059 PUSHAB DIMVECPTR 0C AE 9F 0005C PUSHAB NDIMS 14 AE 9F 0005F PUSHAB CELLTYPE	4203 4204 3598 3615
	000000006 00 52 50 01 50 06	1C AE 9F 00062 PUSHAB DSCADDR 14 AC DD 00065 PUSHL TYPEID 06 FB 00068 CALLS #6, DBG\$STA_TYP_ARRAY 0C AC DO 0006F MOVL SYMID, R2 38 13 00073 BEQL 6\$ 52 DO 00075 MOVL R2, RSTPTR 14 AO 91 00078 4\$: CMPB 20(RSTPTR), #1 06 13 0007C BEQL 5\$ 10 AO DO 0007E MOVL 16(RSTPTR), RSTPTR F4 11 00082 BRB 4\$ 29 AO 91 00084 5\$: CMPB 41(RSTPTR), #6 23 12 00088 BNEQ 6\$	3631 3638 3639 3640 3641
	0000000G 00	52 DD 0008A PUSHL R2 01 FB 0008C CALLS #1, DBG\$STA_SETCONTEXT 10 AE 9F 00093 PUSHAB VALUE_KIND 34 AE 9F 00096 PUSHAB DESC 52 DD 00099 PUSHL R2 03 FB 0009B CALLS #3, DBG\$STA_SYMVALUE 10 AE D1 000A6 PNED 6\$	3644
	00000000G 00 03 0C AE 59 14	03 FB 0009B	3646 3648 3652

BGPARSER 04-000							1	Sep- Sep-	1984 02:10: 1984 12:17:	:13	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page 1
	51	0000000G	00	00028EA8	8F 01 05	DD	000B6 000BC 000C3	70.	PUSHL	#167	592 LIB\$SIGNAL R9, R1	1.
	,,	000000006	00	0A	A1	9F	000C7	7\$:	PUSHAB	111111		36
			55	18	50 A5	FB DO 9E	000D1 000D4		MOVAB	RÓ. 24 (N	NODEPTR ODEPTR), RO	36
		03	A0 52 0E	0C 03	AE AE	90 00 91	80000 0000C		MOVB MOVL	R9, DSCA	DBG\$GET_TEMPMEM NODEPTR ODEPTR), RO 3(RO) DDR, R2), #14	36
		02	AO	03	01 05 05 02 02 05 05 05 05 05 05 05 05 05 05 05 05 05	12	000C7 000CA 000D1 000D8 000DC 000E0 000E4 000EA		PUSHL CALLS MULL3 PUSHAB CALLS MOVL MOVAB MOVB MOVB BNEQ BNEQ BNEQ BNEQ BNEQ BNEQ BNEQ BNE	8\$ #34.	2(RO)	36
		02 10		02	05 A2	11 90	000EA	8\$: 9\$:	BRB	9\$ 2(R2), 2(RO)	
50	04 42	10	A0 A5 60 01	08	62 A2	BO BO EF	000EC 000F1 000F5 000F9 000FF 0010A	9\$:	MOVW	(R2) 8(R2	28(NODEPTR)), (RO)	36 36 36 36
0A A5	0A A2 01	24	01	08	50 AE	FO	000FF 00105		INSV	RO.	#1, #1, 10(NODEPTR) TYPE, 36(NODEPTR)	
			A5 53 5A	08 14	6349	9E DE 8F	0010A 0010E		MOVAB MOVAL	20(R (STR	2), STRIDEPTR IDEPTR)[R9], BOUNDVEC	36 36 36 36
002B 001C 009C	001C 001C 001C	00	010	03	001C	8F	0010E 00112 00117	10\$:	.WORD	3(R2), #1, #13 10s,-	: 36
0090	0016	00	01 01 01 01 09 09 098		001C 001C 001C 001C		0011F 00127 0012F			115-), 2(R0) , 28(NODEPTR)), (R0) #1, #1, 10(R2), R0 #1, #1, 10(NODEPTR) TYPE, 36(NODEPTR) 2), STRIDEPTR IDEPTR)[R9], BOUNDVEC), #1, #13 10\$,- 10\$,- 10\$,-	
										115-	10\$,-	
										446	10\$,- 10\$,-	
										235-	10\$,- 10\$,- 10\$,- 10\$,- 10\$,- 10\$ 248 LIB\$SIGNAL	
										235-	10\$ 10\$	
		0000000G	00	00028198	8F 01 71	DD	00133 00139 00140	11\$:	PUSHL CALLS BRB	#164	248 LIB\$SIGNAL	37
			50	ОС	71 A8 17	11 00	00142	12\$:	MOAF	23\$ 12(R	8), TMP_SYMID	37
			01	14	AO	91	00146	13\$:	CMPB	122	MP_SYMID), #1	37 37 37
			50	10	A0 06 A0 F4	DÖ	0014E 00152		MOVL BRB	153	MP_SYMID), TMP_SYMID	37
	.,	20	05	29	A0 05	91	00154 00158	14\$:	CMPB BNEQ	41 (TI	MP_SYMID), #5	37
	54	28 00000000G	A0		52 01	EO DD	00148 0014C 0014E 00152 00154 00158 0015F 00161	15\$:	PUSHL	D 2	40(TMP_SYMID), 23\$	37 37
	50		51	07	A0 08	FB 9E C7	UUIIDA		BEQL CMPB BEQL MOVL BRB CMPB BNEQ BBS PUSHL CALLS MOVAB DIVL3 MOVAB	7(RO	DBG\$DATA_LENGTH), R1 R1, LENGTH 9), R1 10(R2), 18\$ TH, STRIDE	37
	18		AZ AE	, FF	A0 08 A9 05 50 0A	9E E1	0016C 00170 00174 00179 0017D 0017F		MOVAB BBC	-1 (R	9), R1 10(R2), 18\$	37 37 37
		30	AE		50	D0 D4	0017D		BBC MOVL CLRL BRB	I 17\$	IN, SIKIDE	37

						B 9 16-Sep- 14-Sep-	1984 02:10: 1984 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 146 (19)
3C AE4	0 38	AE40 50	FC	A340 51	C5 0018	1 16\$: B 17\$:	MULL3 AOBLEQ	-4(STRIDEPTR)[I], STRIDE-4[I], STRIDE[I] R1, I, 16\$ 21\$: 3737
	30	AE41 50	FF	18 50 A9 0A	C5 0018 F3 0018 11 0018 D0 0019 9E 0019	1 18\$:	BRB MOVL MOVAB	LENGTH, STRIDE[R1] -1(R9), I	3732 3747 3748
3C AE4	0 40	AE 40 F3 53	04 30	A340	11 0019 C5 0019 F4 001/ 9E 001/	C 195:	BRB MULL3 SOBGEQ MOVAB	20\$ 4(STRIDEPTR)[I], STRIDE+4[I], STRIDE[I] 1, 19\$ STRIDE, STRIDEPTR	3749
	0A	A5 54	28	50 AE 04 04 A5 5B 01	11 001A	D 225:	BRB BISB2 MOVAB	23\$ #4, 10(NODEPTR) 40(R5), SUBVECTOR	3690 3774 3793
		52		5B 01	9E 001E D4 001E CE 001E 31 001E C5 001E	7	CLRL	WI. I	3794
5	7	52 52		009D 14 01	78 0010	.5	BRW MULL3 ASHL	29\$ #20. I. R7 #1, I. R6	3797
		9E	04	6744 6846 8744	9F 0010 9F 0010 D0 0010	A	PUSHAB MOVL PUSHAB	#20, I, R7 #1, I, R6 (R7)[\$UBVECTOR] (BOUNDVEC)[R6], a(SP)+ 4(R7)[\$UBVECTOR] (STRIDEPTR)[I], a(SP)+ 8(R7)[\$UBVECTOR] (BOUNDVEC)[R6], a(SP)+ 12(R7)[\$UBVECTOR] 4(ROUNDVEC)[R6], a(SP)+	3798
		9E 9E	08	6342 A744 6A46	DO 0010 9F 0010 DO 0010)6	MOVL PUSHAB MOVL	(STRIDEPTR)[I], a(SP)+ 8(R7)[SUBVECTOR] (BOUNDVEC)[PA] a(SP)+	3799
		9E	0C 04	A744 AA46	9F 0010	SE SE	PUSHAB MOVL	12(R7)[SUBVECTOR] 4(BOUNDVEC)[R6], a(SP)+ SUB_TYPEID	3800
		50	90	BE42	D4 001E D0 001E 13 001E	A	MOVL CLRL MOVL BEOL	aDIMVECPTR[I], RO	3802 3803
	20	AE 50 09	20 18	2F 50 AE AO 17	DO 001F DO 001F 91 001F	5 25\$:	BEQL MOVL MOVL CMPB	RO, SUB TYPEID SUB TYPEID, RO 24(RO), #9	3806 3807
			14 10 24 20	AE AE AE	12 001f 9F 001f 9F 0020 9F 0020)2)5	BNEQ PUSHAB PUSHAB PUSHAB PUSHAB	26\$ SIZE HIGHPTR LOWPTR SUB_TYPEID	3808
	00000000	00		AE 50 05 DF	DD 0020 FB 0020 11 0021	08 08 00 14 16 26\$:	PUSHL CALLS BRB	#5 DBG\$STA_TYP_SUBRNG	
		9E	10	A744 AE 07	9F 0021 D0 0021 11 0021	6 26\$:	BRB PUSHAB MOVL	16(R7)[SUBVECTOR] SUB_TYPEID, a(SP)+ 28\$	3810
		9E	10	A744	9F 0022	275:	BRB PUSHAB MOVL	1A(D/) SIIRVECTOD	3803 3814
5	6		04	A744 6A46	DO 0022 9F 0022 C5 0022 C2 0023	7 28\$:	MOVL PUSHAB MULL3	(BOUNDVEC)[R6], a(SP)+, R6	3817
		9E 5B 09	00000000	56 20 AE 1A	91 0023	53	CMPB BNEQ	RO, a(SP)+ 4(R7)[SUBVECTOR] (BOUNDVEC)[R6], a(SP)+, R6 R6, OFFSET DBG\$GB_LANGUAGE, #9 29\$	3816 3825
		50	20	AE 1A	DO 0023	0	MULL3 SUBL2 CMPB BNEQ MOVL BEQL CMPB	29\$ TYPEID, RU	3827
		04	18	A0 14 6744	91 0024 12 0024 9F 0024	.8	BNEQ PUSHAB	24(RO), #4 29\$ (R7)[SUBVECTOR]	3829
	00000000	00		9E 50 02	DD 0024 DD 0024 FB 0024	B	PUSHL PUSHL CALLS	a(SP)+ RO #2, DBG\$ENUM_VAL	3832

DB VO

DBG\$GET_TEMPMEM NODEPTR

FCODE

CALLS

MOVL

BRW

MOVL

0000000G

10

DB

GPARSER 4-000								16-Sep-1	1984 02:10 1984 12:17):13 7:30	VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1	Page 14 (19
			0000000G	00		0F 01	DD FB	0031D 0031F	PUSHL	#15	DBG\$GET_TEMPMEM	: 393
				55	18	50	90 9E 9E	0031D 0031F 00326 00329	PUSHL CALLS MOVAB MOVAB MOVAB EXTZV MOVW CMPZV BNEQ BITB BEQL MOVB BRB	RO.	DBG\$GET_TEMPMEM NODEPTR NODEPTR), R1 3(R1) R2), R0 #4, (R0), R3 28(NODEPTR) #4, (R0), #1	393
53		60	03	50	0A	A2	9E	0032D 00331	MOVAB	100	3(R1) R2), R0	394
01		60	10	A5 04		53	BO	0033A 0033F	MOVW	#0. R3. #0.	28(NODEPTR) #4 (RO) #1	394
			FO	8F		A0A0502066C260206733D0201608A440	12	00343	BNEQ			395
			02	A1		06	13	00345 00349 0034B 0034F	MOVB	43\$	2(R1)	395
			02	A1		92	90	00351 435:	MOVB	#6, 51\$ #2, 51\$	2(R1)	395
02		60		04		00	ED 12	00355	BRB	#0,	#4, (RO), #2	395 395 395
			FO	8F		60	93	0035E	BNEQ BITB BEQL MOVB BRB MOVB	(RO)	. #240	396
			02	A1		07 33	90	00364 00368	MOVB	#7. 51\$	2(R1)	396
			02	A1		03	90	0036E	MOVB BRB CMPZV	51\$	#4, (R0), #2), #240 2(R1) 2(R1) #4, (R0), #4	396 396 396
04		60		04		12	ED 12	00370 46\$: 00375	CMPZV BNEQ	48\$	#4, (R0), #4	
			F0	8F		06	93	0037B	BNEQ BITB BEQL MOVB	475	, #240	397
			02	A1		1A	11 90	00381 00383 47\$	BRB MOVB	#8, 51\$ #4, 51\$	2(R1) 2(R1)	397
			FO	8F		14	93	00387 00389 48\$:	BRB BITB	51\$ (RO)		397 397 398
			02	A1		06	13 90 11	0038D 0038F	BEQL MOVB	49\$	2(R1)	398
			02 10	A1		22	90	00393 00395 49\$:	BRB MOVB	50\$	2(R1)	
04 A4		40		A1 A5 54 04	28	A5	90 A4 9E EF	00399 50\$: 00390 51\$:	MOVAB	40(1	28(NODEPTR) R5), SUBVECTOR	399
04 A4		60	10	AC		01 04 208 A5 02 01	11	003A7 003A9 52\$:	BRB	53\$	FCODE	398 399 400 401 402
			00000000	00		OF	DD FB	003AD 003AF	PUSHL	#15	DBG\$GET_TEMPMEM	402
			OA 1A	55 A5		50	00 88	003B6 003B9	MOVL BISB2	RO.	NODEPTR 10(NODEPTR)	403
				A5 54	00010122	8F A5	00 88 90 90 05	003BD 003C5	MOVAB	40(326, 26(NODEPTR) R5), SUBVECTOR	403 403 403 404
			04	A4	06	01 50 8F A5 01 A2 3A	D5	00377 0037B 0037D 00381 00383 00387 00389 00386 00399 00395 00399 00395 00397 003A7 003A7 003A7 003A7 003A7 003A8 003B6 003B6 003B9	BEGL MOVB BRB MOVAB MOVAB EXTZV BRB MOVL PUSHL CALLS MOVL BISB2 MOVL MOVAB MOVAB MOVL SUBL3 BRB BISB2 MOVL CALLS	6(R	2(R1) 2(R1) 28(NODEPTR) 25), SUBVECTOR 44, (R0), 4(SUBVECTOR) FCODE DBG\$GET_TEMPMEM NODEPTR 10(NODEPTR) 36, 26(NODEPTR) 35), SUBVECTOR 4(SUBVECTOR) 2) 6(R2), 12(SUBVECTOR) (R3) FCODE DBG\$GET_TEMPMEM	404
	00	A4	06	A2		01	C3	003D2 003D8	SUBL3	575	6(R2), 12(SUBVECTOR)	405
			10	63 AC		10 01 0F 01	88 00 00 FB	003DA 548:	BISB2 MOVL	#16	FCODE	405 391 406 406
			0000000G	00		0F 01	DD FB	003E1 003E3	PUSHL	#15	DBGSGET TEMPMEM	406

DB

645:

#^M<R2,R3>

53

V

DBGPARSER V04-000			F 9 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 150 (19)
	00000000G 00 24 A5 10 52 53	10	15 12 004BA BNEQ 66\$ 08 DO 004BC MOVL #8, ATOMIC_TYPE 20 DO 004BF MOVL #32, BIT_LENGTH 7E D4 004C2 CLRL -(SP)	4187 4190 4191 4192
	00000000G 00 14 AC 08 A5 09 A5 0C A5 10 A5 07 A8 06 A8 08 A8 18 B8	08 10 14 00 14 08 10 14	0C BB 004C4 PUSHR #^M <r2,r3> 03 FB 004C6 CALLS #3, DBG\$TYPEID_FOR_ATOMIC 50 D0 004CD MOVL RO, TYPEID AC 90 004D1 66\$: MOVB KIND, 8(NODEPTR) AC D0 004DB MOVL TYPEID, 12(NODEPTR) AC D0 004E0 MOVL SYMID, 16(NODEPTR) AC D0 004E5 CLRL 20(NODEPTR) AC 90 004E8 MOVB KIND, 7(R8) AC 90 004EB MOVB KIND, 7(R8) AC 90 004ED MOVB FCODE, 6(R8) AC 90 004F2 MOVL TYPEID, 8(R8) AC D0 004F2 MOVL TYPEID, 8(R8) AC D0 004F7 INSQUE (NODEPTR), a24(R8) O4 004FB</r2,r3>	4212 4213 4214 4215 4216 4224 4225 4231 4234

; Routine Size: 1276 bytes, Routine Base: DBG\$CODE + 006C

VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32:1 Page 151

V

GLOBAL ROUTINE DBG\$EXP_INT(INPUT_DESC, RADIX, VALUE_PTR, TERM_INDEX) =

FUNCTION

This is the common Expression Interpreter for most languages supported by DEBUG. It parses and evaluates a source language expression and returns a Value Descriptor which represents the value of the expression. This routine itself is only a set-up routine which sets up the character pointer and the expression radix to use and then calls DBG\$EXPRESSION_PARSER to do the actual work.

INPUTS

INPUT_DESC - The address of a VAX standard string descriptor which describes the input string to be parsed. The length is actually not used, however--the string is instead assumed to be terminated by a carriage-return character.

RADIX - The radix to be used to interpret integer constants in the input string. The allowed radix values are DBG\$K_DECIMAL, DBG\$K_HEX, DBG\$K_OCTAL, and DBG\$K_BINARY.

VALUE_PTR - The address of a longword to receive a pointer to the value descriptor returned by this routine as its output.

TERM_INDEX - A "terminator index" which indicates which lexical tokens are allowed as expression terminators in this context. These index values have names of the form TOKEN\$K_TERM_xxx.

5th Optional Parameter - If this is present, and the value is DBG\$K DEPOSIT VERB then pass this into DBG\$EXPRESSION_PARSER, so that in DBG\$EXPRESSION_PARSER, when the expression is not address expression and in deposit command, DBG\$EVAL LANG OPERATOR will not be called with DBG\$GL_IDENTITY_TOKEN.

(This is passed in from DBG\$NPARSE_DEPOSIT, from DBG\$NPARSE_EXPRESSION).

OUTPUTS

VALUE_PTR - The address of a Value Descriptor is returned to VALUE_PTR.
This Value Descriptor represents the value of the expression interpreted by DBG\$EXP_INT.

The value STS\$K_SUCCESS is return as the routine result if the expression was terminated by a carriage-return character. If it was terminated any other way (i.e., by a terminator token), the value STS\$K_WARNING is returned.

BEGIN

MAP

LOCAL

VALPTR: REF DBG\$VALDESC;

! Pointer to returned Value Descriptor

24

5

5

```
9
DBGPARSER
V04-000
                                                                           16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                       VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32;1
  BUILTIN ACTUALCOUNT, ACTUAL PARAMETER:
                                   Set up CHARPTR to point to the start of the expression string. Also set
                                   up the radix we are to use in the scan and initialize some variables.
                                 CHARPTR = .INPUT_DESC[DSC$A_POINTER];
EXPRESSION_RADIX = .RADIX;
                                 SAVED_TOKEN = 0;
                                   Call the Expression Parser to parse the language expression. If the
                                   result is an unconverted constant Value Descriptor, we convert it to
                                   a real Value Descriptor here. Then return the Value Descriptor pointer.
                                 IF ACTUALCOUNT() GTR 4
                                 THEN
                                     BEGIN
                                      IF ACTUALPARAMETER(5) NEQ DBG$K_DEPOSIT_VERB
                                      THEN
                                          $DBG_ERROR('DBGPARSER\DBG$EXP_INT');
                                     VALPTR = DBG$EXPRESSION_PARSER(FALSE, .TERM_POINTER_TBL[.TERM_INDEX] + TABLEBASE,
                                                                 DBG$K_DEPOSIT_VERB);
                                     END
                                ELSE
                                     VALPTR = DBG$EXPRESSION_PARSER(FALSE,
                                                                  .TERM_POINTER_TBL[.TERM_INDEX] + TABLEBASE);
                                VALUE_PTR[0] = .VALPTR;
                                   fix up the string descriptor to reflect the new location of the parse
                                   pointer and return the appropriate status code.
                                INPUT_DESC[DSC$W_LENGTH] = .INPUT_DESC[DSC$W_LENGTH] + .INPUT_DESC[DSC$A_POINTER] - .CHARPTR;

INPUT_DESC[DSC$A_POINTER] = .CHARPTR;

IF .CRARPTR[O] NEQ CAR_RET THEN RETURN STS$K_WARNING;
                                 RETURN STS$K_SUCCESS;
                                 END:
                                                                                       .PSECT
                                                                                                DBG$PLIT, NOWRT, SHR, PIC, O
   47 42 44 50 52 45 53
                                                                      03040 P.AWR:
                                                                                       .ASCII
                                                                                                <21>\DBGPARSER\<92>\DBG$EXP_INT\
                                                                                       .PSECT
                                                                                                DBG$CODE, NOWRT, SHR, PIC, O
                                                                003C 00000
                                                                                                DBG$EXP_INT, Save R2,R3,R4,R5
                                                                                                                                                    : 4235
                                                                                       .ENTRY
```

DBGPARSER V04-000				1	1 9 6-Sep-1984 02:10 4-Sep-1984 12:17		Page 153 (20)
	040c C5	0424	EF AS AC AC AC AC AC AC AC AC AC AC AC AC AC	9E 00002 9E 00009 D0 00010 D0 00018 D4 0001E D0 00022 91 00026 1B 00029	MOVAB MOVAB MOVL MOVL CLRL MOVL CMPB BLEQU	CHARPTR, R5 TERM_POINTER_TBL, R4 INPUT_DESC, R3 4(R3), CHARPTR RADIX, EXPRESSION_RADIX SAVED_TOKEN TERM_INDEX, R2 (AP), #4	4299 4300 4301 4316 4308
	05		AC 13	DI UUUED	CMPL	2\$ 20(AP), #5 1\$	4311
	000000006 00	2A6C 00028362	01 8F 03	9F 00031 DD 00035 DD 00037 FB 0003D	CMPL BEQL PUSHAB PUSHL CALLS PUSHL CALLS MOVL MOVAB PUSHAB	P.AWR #1 #164706 #3, LIB\$SIGNAL	4313
	50 51	FAAF	6442	DD 00046 DO 00046 9E 0004A 9F 0004F D4 00052 FB 00054	1\$: PUSHL MOVL MOVAB PUSHAB	TERM POINTER TBL[R2], RO TABLEBASE, RT (R1)[R0]	4315
	0000V CF		03	PB 00054	CLRL CALLS BRB	-(SP) #3, DBG\$EXPRESSION_PARSER 3\$	4315
	50 51	FAAF	6442	9E 0005F	25: MOVL MOVAB	TERM_POINTER_TBL[R2], RO TABLEBASE, RT (R1)[R0]	4308
	0000V CF 0C BC 51 51	04	7E 050 633 650 500 604 01	D4 00067 FB 00069 D0 0006E 3C 00072 C0 00079 A3 00070 D0 00080	CLRL CALLS MOVL MOVZWL ADDL2 MOVL	-(SP) #2, DBG\$EXPRESSION_PARSER VALPTR, aVALUE_PTR (R3), R1 4(R3), R1 CHARPTR, R0	4321 4323 4330
63			50 50 60 04	A3 0007C D0 00080 91 00084 12 00087 D0 00089	MOVL SUBW3 MOVL CMPB BNEQ	RO, R1, (R3) RO, 4(R3) (RO), #13 4\$	4331 4332
	50		01 50	00 00089 04 00080 04 00080 04 0008F	MOVL RET	#1, RO RO	4333 4335
. Pouting Cine. 1// butes	Dautina Da	DDCE	-	0549			

; Routine Size: 144 bytes, Routine Base: DBG\$CODE + 0568

Page 154

GLOBAL ROUTINE DBG\$EXPRESSION_PARSER(ADDRESS_EXPRESSION, TERM_LIST) =

FUNCTION

This routine parses and interprets either a DEBUG Address Expression or a language expression in the current language and returns the result of the expression evaluation.

The routine uses an Operator Precedence parsing scheme. Each operator is represented by an Operator Lexical Token Entry which contains the kind of the operator (prefix, infix, or postfix) and the left and right precedences of that operator. Operands are represented by Primary Descriptors, Value Descriptors, or other Operand Lexical Token Entries. The operators and operands are retrieved by calling the Primary Parser.

When an operand is encountered, it is simply stacked on the operand stack. When an operator is encountered, its left precedence is compared to the right precedence of the previous operator on the operator stack. If the previous operator has the higher or equal precedence, it is popped from the operator stack and evaluated. The evaluation requires one or two operands to be popped from the operand stack, after which the result is pushed back on that stack. When no previous operator has a higher or equal precedence, the new operator is pushed onto the operator stack.

The operator stack is always initialized with the "initiator operator" which ensures that there is always a previous operator on the stack. The end of an expression is always signalled by the "terminator operator" whose left precedence is set such that it forces evaluation of all operators still on the operator stack up to the initiator operator. The single operand left on the operand stack thereafter constitutes the result of the expression evaluation.

This routine accepts a list of allowed "terminator tokens" (keywords such as "DO" or "THEN" or special characters such as ",", ")", or "=", depending on context). This list is passed to the Lexical Scanner which returns the Terminator Operator when such a token or a carriage-return is encountered. As a side effect, OWN variable TERMINATOR CODE is set to a value which indicates which terminator token was found. That terminator's character length is also set in TERMINATOR LENGTH. (This side effect is used when parsing subscript expressions.)

INPUTS

ADDRESS_EXPRESSION - A flag set to TRUE if a DEBUG Address Expression is to be parsed and evaluated. If this flag is FALSE, a language expression for the current language is parsed and evaluated instead. This flag affects both the lexical scanning of operator symbols and the parsing and evaluation of the expression operators.

TERM_LIST - A vector of pointers to Terminator Lexical Token Entries for the Terminator Tokens which can terminate the expression to be parsed. The vector must be in PLIT form (TERM_LIST[-1] gives the number of entries) and each pointer is expected to be relative to TABLEBASE. If there are no terminator tokens other than carriage return, this list is empty (0 entries).

3rd Optional Parameter - If this is present, and the value is

VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1

```
DBGPARSER
V04-000
                                                                                                                  VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                                              [TOKENSK_RADIX_BIN]:
EXPRESSION_RADIX = DBG$K_BINARY;
                                                               ! Infix NOT should not show up here.
                                                               LTOKENSK_INFIX_NOT]:
                                                                    SIGNAL (DBG$_MISOPEMIS, 1, LEFT_OP[TOKEN$9_OPLEN]);
                                                                Any other case is an error--a lexical operator should not be on the stack or should not be marked as lexical
                                                                 if it has no semantic action here.
                                                              COTHERWISE]:
                                                                    $DBG_ERROR('DBGPARSER\EXPRESSION_PARSER 20');
                                                              TES:
                                                         END
                                                      This is not a lexical operator -- it is a "normal" operator.
                                                      If this is a DEBUG Address Expression, evaluate the Address
                                                      Expression operator.
                                                    ELSE IF .ADDRESS_EXPRESSION
                                                         OPERAND_STACK[.OPAND_INDEX] = DBG$EVAL_ADDR_OPERATOR(.LEFT_OP, .LEFT_ARG, .RIGHT_ARG)
                                                      And if it is a language expression, evaluate this non-lexical
                                                      operator according to language rules.
                                                    ELSE
                                                         BEGIN
                                                          EVAL_LANG_OPERATOR returns a pointer to a Primary Descriptor
                                                           or a pointer to a Value Descriptor.
                     4660
                     4661
                                                         PRIMPTR = DBG$EVAL_LANG_OPERATOR(.LEFT_OP, .LEFT_ARG, .RIGHT_ARG);
                                                           If a Primary Descriptor was returned from the expression evaluator then there may be some further processing to do. An example of this is the C expression:
                                                            EVALUATE (*PTR).COMPONENT
                                                           The (*PTR) will be evaluated by the expression evaluator and a Primary Descriptor will be returned (presumably,
                                                           describing a record). Then we need to call the Primary Parser to pick up the rest of the expression.
                                                             .PRIMPTR[DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC
                                                         THEN
                                                              BEGIN
                                                              DBGSPRIMARY_PARSER (
                                                                   FALSE.
                                                                                              ! Operand not expected
```

DE

```
C 10
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                           VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                                                        FALSE,
TERM_LIST,
  Not address expression
                                                                 Pass along te Parenthesis not used PRIMPTR, Not used Input Primary REMEMBER C STATE GOT SUBSCRIPT);
PRIMPTR = .NEW_PRIMPTR;
END;
                                                                                                       Pass along terminator list
Parenthesis nesting
Address to fill in result
                                                                Now just put the result of the expression on top of
                                                                the expression stack.
                                                              OPERAND_STACK[.OPAND_INDEX] = .PRIMPTR;
                                                             END:
                                                        END:
                                                                                         ! End of pop and evaluate loop
                                                     If the new operator is a "lexical operator", meaning that it has some semantic effect on the lexical processing or parsing of the
                                                     current expression, we perform that semantic action here. For
                                                     example, the terminator operator has the semantic effect of term-
                                                     inating the scan of the current expression.
                                                  IF .TOKEN[TOKEN$V_LEXICAL]
THEN
                                                        BEGIN
                                                        SELECTONE .TOKEN[TOKEN$W_CODE] OF
                                                                If this is the terminator operator, exit the parse loop.
                                                             TOKENSK TERMINATOR]:
EXITCOOP;
                                                               If this is an open parenthesis "(", increment the
                                                                parenthesis count.
                                                             [TOKEN$K_OPENPAREN]:
                                                                   PAREN_NESTING = .PAREN_NESTING + 1;
                                                                If this is a close parenthesis ")", decrement the parenthesis count, check for balanced parentheses, and remove
                                                                both the close parenthesis and the matching open paren-
                                                                thesis from the operator stack.
                                                             TOKENSK CLOSEPAREN]:

BEGIN

PAREN_NESTING = .PAREN_NESTING - 1;

IF .PAREN_NESTING LSS 0 THEN SIGNAL (DBGS_UNBPAREN);

IF .OPTOR_INDEX LSS 1
                                                                         $DBG_ERROR('DBGPARSER\EXPRESSION_PARSER 2');
```

DE

Page 160 (21)

V

DE

DV

```
6 10
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                                                                                                                                                                         VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32;1
     4795

479678

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

479978

                                                  Stack the current operator on the operator stack. Then loop to
                                                                                                                          get the next operator or operand.
                                                                                                                  OPTOR INDEX = .OPTOR INDEX + 1;
IF .OPTOR_INDEX GEQ MAX_OPTOR_INDEX THEN SIGNAL(DBG$_PARSTKOVR);
OPERATOR_STACK[.OPTOR_INDEX] = .TOKEN;
                                                                                                                  END:
                                                                                                                                                                                                            ! End of ELSE-clause for operators
                                                                                                     END:
                                                                                                                                                                                                            ! End of the get-symbol loop
                                                                                               We are all done parsing the expression. Retrieve the descriptor from
                                                                                               the top of the stack.
                                                                                         if .OPAND_INDEX GTR 0 THEN $DBG_ERROR('DBGPARSER\EXPRESSION_PARSER 3');
valptr = .OPERAND_STACK[0];
                                                                                              If this is a language expression, then we always return a Value Descriptor. Primary Descriptors or Volatie Value Descriptors are converted to Value Descriptors here. DBG$EVAL_LANG_OPERATOR does this for us.
                                                                                         IF NOT .ADDRESS_EXPRESSION AND NOT .DEPOSIT_FLAG
                                                                                         THEN
                                                                                                     VALPTR = DBG$EVAL_LANG_OPERATOR (DBG$GL_IDENTITY_TOKEN, .VALPTR, 0);
                                                                                               If this is an address expression then we always return either a
                                                                                               Primary Descriptor or a Volatile Value Descriptor. DBG$EVAL_ADDR_OPERATOR does this for us.
                                                                                                 .ADDRESS_EXPRESSION
                                                                                                     VALPTR = DBGSEVAL_ADDR_OPERATOR (DBGSGL_IDENTITY_TOKEN, .VALPTR, 0);
                                                                                         RETURN . VALPTR;
                                                                                        END:
                                                                                                                                                                                                                                             .PSECT
                                                                                                                                                                                                                                                                    DBG$PLIT, NOWRT, SHR, PIC, O
                                                                                                                                                                                                                  P.AWS:
                                                                                                                                                                                                                                            .ASCII
                                                                                                                                                                                                                                                                     <31>\DBGNPARSE\<92>\DBG$NPARSE_EXPRESS\
                                                                                                                                                                     440
473
43
                                                                                                                                                                                  499051
                                                                                                                                                                                                                                                                     \ION\
                                                                                                                                                                                                                                                                     <9>\somewhere\
                                                                                                                                                                                                                  P.AWT:
                                                                                                                                                                                                                 P.AWU:
                                                                                                                                                                                                                                                                     <29>\DBGPARSER\<92>\EXPRESSION_PARSER \
52
                                                                                                                                                                                                                                                                     <30>\DBGPARSER\<92>\EXPRESSION_PARSER \
                                                                                                                                                                                                                                                                     \20\
<29>\DBGPARSER\<92>\EXPRESSION_PARSER \
```

DBGI VO4-	PARSI	ER													1	10 5-Sep-19 4-Sep-19	084 02:10 084 12:17	:13 VAX-11 Bliss-32 V4.0-742 Pa :30 [DEBUG.SRC]DBGPARSER.B32;1	ige 165 (21)
52	50	58	45	ŞÇ	52	45	53	52 5F	41	50	47	42	44	32 10	030DA 030DB	P.AWX:	:ASCII	\2\ <29>\DBGPARSER\<92>\EXPRESSION_PARSER \	:
52													44	10 45 33 10	030F8 030F9	P.AWY:	.ASCII	\3\ <29>\DBGPARSER\<92>\EXPRESSION_PARSER \	
ā	50	52	45	53	52	41	50	52 5F	4E	4F	49	53	53	33	030F8 030F9 03108 03116		.ASCII		
																	.PSECT	DBG\$CODE,NOWRT, SHR, PIC,0	
														OFFC	00000		.ENTRY	DBG\$EXPRESSION_PARSER, Save R2,R3,R4,R5,R6,-R7,R8,R9,R10,RT1 -232(SP), SP	: 4336
										SE.	-	F18	CE	9E	00002		MOVAB	-232(SP), SP	: ,,,,
										02			7E 6C	91 91	00009		CLRL	DEPOSIT FLAG	: 4449
										05		00	AC	18 01 13 9F 0D 0D FB	0000E 00012		BLEQU	12(AP), #5	: 4453
										(00000	0000	EF	9F	00014		PUSHAB	P. AWS	: 4455
							000	0000	06	00	00028	3362	8F 03 01 56	DD	0001A 0001C 00022		PUSHL PUSHL CALLS	#164706	
							000	0000	00	00 6E			01	DO	00029	15:	MOVL	#3, LIB\$SIGNAL #1, DEPOSIT_FLAG	4457
								2	4 A	46 (00000	0000	ÉF	9E	0002C 0002E	2\$:	MOVAB	#1, DEPOSIT_FLAG OPTOR_INDEX INITIATOR_TOKEN, OPERATOR_STACK-	: 4465
								0	,	S7 AE			01	CE	00037		MNEGL	M1. OPAND INDEX	: 4467
								U	•	ME		14	O1 5B AE AE	00 04 9F 9F	00037 0003A 0003E 00040 00043	70.	CLRL	#1. OPERAND EXPECTED PAREN_NESTING	: 4468 : 4469 : 4485
												16	AE	9F	00040	3\$:	PUSHAB	TOKEN_OPERAND_FLAG	:
										7E		04 18	AC	7D	00048		PUSHL	ADDRESS_EXPRESSION, -(SP)	: 4486 : 4485
								000	00	CF			06	FB	0004E		PUSHL CALLS MOVL	#6, DBGSPRIMARY_PARSER	
										58 46 15		18	AE	E9	00058		BLBC	TOKEN OPERAND FLAG, 6\$: 4504
										15 (00000	0000.	SB AC AE O6 AE AE EF	DD FB00 FB4 D61 D19	00050		BLBC BLBS PUSHAB PUSHL PUSHL CALLS	PAREN_NESTING ADDRESS_EXPRESSION, -(SP) OPERAND_EXPECTED #6, DBG\$PRIMARY_PARSER TOKEN, R8 TOKEN_OPERAND_FLAG, 6\$ OPERAND_EXPECTED, 4\$ P.AWT	: 4496
							000	0000	0.0	(00028	B9AA	8F	DD	00068		PUSHL	#166314	
							000	0000	UG	00		04	8F 03 AE 57	04	00065	48:	CLRL	#3, LIB\$SIGNAL OPERAND EXPECTED OPAND_INDEX OPAND_INDEX, #25	4501
										19			57	D1	00078		CMPL	OPAND_INDEX, #25	4501 4502 4503
											00028	BOEO	8F	00	0007b		BLSS PUSHL CALLS		
								0000	C AL	00			58	DD FB DD E1 DD FB	00085	58:	MOVL	#164064 #1, LIB\$SIGNAL R8, OPERAND_STACK[OPAND_INDEX] #3, DBG\$GL_DEVELOPER, 3\$ R8	4504
						A7	000	0000		00			58	DD	00091		MOVL BBC PUSHL CALLS	#3. DBG\$GL_DEVELOPER, 3\$: 4505
								000	OV	CF			0D 8F 01 58 03 58 01 9E 8E	FB 11	00046 00048 0004F 00058 00056 00066 00068 00078 00078 00078 00078 00078 00085 00085 00085 00085		BRB	#1, DUMP_PRIMARY	: 4493
										0A 02		04 18	AE BE	E9	000A2	65:	BRB BLBC CMPB	OPERAND_EXPECTED, 7\$ atoken, #2	4493 4525 4526

						1	1 10 5-Sep-1 4-Sep-1	984 02:10 984 12:17	:13 :30	VAX-11 Bliss-32 V4.0-742 CDEBUG.SRCJDBGPARSER.B32;1	Page 166 (21)
		1A 02	04	OA AE BE 14	12 E8 91	000AA 000AC 000B0	78:	BNEQ BLBS CMPB	8\$ OPERAN aTOKEN	ND_EXPECTED, 9\$	4527
7E	18	AE		14	12	000B4 000B6	85:	BNEQ ADDL3	9\$	TOKEN, -(SP)	4530
		-	000289B2	00 01 8F 03 68 04	DD DD FB	000BB 000BD		PUSHL	#1 #16632		. 4330
	0000000G	00	00020702	03	FB 91	000C3	oe.	CALLS	#3, L	IB\$SIGNAL	
	•			04	13	000CD	9\$:	CMPB BEQL MOVL	(R8),		4532
	04	AE 59 A8	24	AE46	D0 D0 91	000CF 000D3	10\$:	MOVL	OPÉRAT	PERAND_EXPECTED TOR_STACK[OPTOR_INDEX], LEFT_O T_OP), 5(R8)	P 4534
	05	A8	04	A9	91 1E	80000 00000		CMPB BGEQU	113	T_OP), 5(R8)	4550
		15		0121	31 F4	000DF 000E2	115:	BRW SOBGEQ	28\$	INDEX, 12\$	4550
		.,	00000000	01 8F	9F DD	000E5 000EB		PUSHAB	P. AWU		: 4559 : 4562
	00000000	-	00028362	8F	DD	000ED		PUSHL	441171	06	1
	900000006	00 AE	90		FB DO	000F3 000FA	125:	MOVL	OPÉRAN	IB\$SIGNAL ND_STACK[OPAND_INDEX],	G : 4564
		03	00	AE 69	91	00100		MOVL CLRL CMPB BNEQ	RIGHT.	IB\$SIGNAL ND_STACK[OPAND_INDEX], LEFT_AR ARG OP), #3	: 4565 : 4566
	OC	AE	08	69 0D AE 57	12	00106		BNEQ	13\$	ARG, RIGHT_ARG	4569
	08			57 AD47	D7	0010D 0010F		DECL	OPAND	INDEX	4570
09	000000006	AE 00	70		DO E1	00115	13\$:	MOVL BBC CLRL	#3. DE	BGSGL_DEVELOPER, 14\$	G : 4571
	10,000			03 7E 59 09 A9	D4 DD FB	0011D 0011F		PUSHL	LEFT_C	OP .	
7B	0000v	CF 69		02	FB E1	00121 00126	145:	CALLS	#2. DI	UMP_OPERATOR LEFT_OP) . 24\$	4582
		0B	02	A9 OF	B1	0012A		BBC CMPW BNEQ	2(LEF1	LEFT_OP), 24\$ T_OP), #11	4582 4593
	00000000	00	000289FA	8F	DD	00130		PUSHL	#16639	94	: 4594
	0000000G	00		94	FB 11	00130 00136 0013D 0013F 00143	158:	CALLS BRB	10\$	IB\$SIGNAL	
		34	02	09	B1	0015F 00143	16\$:	BRB CMPW BNEQ	18\$	T_OP), #52	4600
	00000000.	EF		0A	DO 11	00140	175:	MOVI	#10, E	EXPRESSION_RADIX	4601
		35	02	A9		0014E	18\$:	BRB CMPW BNEQ	2(LEF1	T_OP), #53	4607
	00000000.	EF		01 949 008 849 01 04 008 01 04 04 04 04 04 04 04 04 04 04 04 04 04	B1 12 00 11	0014E 00152 00154 0015B 0015D		MOVL	#16. E	EXPRESSION_RADIX	4608
		36	02	A9	81	00150	195:	BRB CMPW	2(LEF1	T_OP), #54	4614
	00000000	EF		09	12 00	00161		BNEQ MOVL BRB CMPW BNEQ	#8, E)	RPRESSION_RADIX	4615
		37	02	51 A9	11 B1	0016A 0016C 00170 00172 00179	20\$:	BRB	25\$ 2(LEF1	T OP) . #55	4621
	00000000	EF		09	12	00170		BNEQ	215	KPRESSION_RADIX	4622
	0000000		02	42	D0	00179	210.	BRB CMPW	25\$		
		50	02	00	B1 12 9F	0017B 0017F 00181 00184	21\$:	BNEQ	22\$	T_OP), #44 FT_OP)	4627
			00	01	9F DD	00181		PUSHAB	#1	1_OP)	4628

DBGPARSER V04-000								1	1 10 5-Sep- 4-Sep-	1984 02:10 1984 12:17	:13	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page 16
					00028982	8F 0E	DD 11	00186 00180		PUSHL BRB	#166 23\$	322	1
					00000000.	EF 01	9F DD	0018F	22\$:	PUSHAB	P.AW	V	463
			0000000G	00	00028362	0E EF 01 8F 03 98	DD DD FB	00194 00196 00190 001A3	23\$:	PUSHL	#164	706 LIB\$SIGNAL	
				16	04	98	- 11	001A3	248:	BRB	15\$	LIB\$SIGNAL ESS EXPRESSION 26\$: 458 : 464
					04 00 00	AC AE AE 59	E9 DD	001A9		PUSHL	RIGH	T ARG	
			000000006	00	•	59	DD	001A5 001A9 001AC 001AF 001B1		PUSHL	LEFT	ESS_EXPRESSION, 26\$ T_ARG _ARG OP DBG\$EVAL_ADDR_OPERATOR OPERAND_STACK[OPAND_INDEX]	
			00000000G 9C AD	47		50 80	00	00188	250.	MOVL	RO.	OPERAND_STACK[OPAND_INDEX]	1,
					0C 0C	AE	DD	001BF	25\$: 26\$:	PUSHL	RIGH	T_ARG	: 464
			00000000	00	oc	AE S9	DD DD	00105		PUSHL	LEFT	OP	
0000079 8F	10	DE	00000000G	00 AE 08		50	FB DO	001CE		MOVL	RO,	PRIMPTR #424	
0000079 8F	10	BE		08		10	12	001DC		BNEQ	27\$	#8, aPRIMPIR, #121	467
					14	AE	DD	001DE 001EQ		PUSHL	PRIM	T_ARG _ARG OP BBG\$EVAL_LANG_OPERATOR PRIMPTR #8, aprimptr, #121 PTR	: 467 : 468 : 467
					14 24 20	16 AE AE AE 7E	9F 9F	001E3 001E6		PUSHAB	NEW	PRIMPTR	: 467
					08	7E AC	D4	00188 0018D 001C2 001C7 001CE 001DC 001DC 001E0 001E8 001EB 001F0 001F0		PUSHAB PUSHL PUSHL CALLS BRB BLBC PUSHL	-(SP	LIST	467
			0000v	CF		7E 08	7C FB	001EE 001F0		CALLS	-(SP	T DBG\$PRIMARY_PARSER	
			10 9C AD	AE 47	20 10	AE	D0	001F5 001FA	275:	MOVL	PRIM	DBG\$PRIMARY_PARSER PRIMPTR, PRIMPTR PTR, OPERAND_STACK[OPAND_INDEX]	: 468
		48		68	F	60 09	DO 31 E1 B1	00200 00203	28\$:	BRW BBC	1112		468 469 454 470 471
				68 02	02	AC 78 AE 09 AS 03	B1 12	001FA 00200 00203 00207 0020B 0020D 00210		BRW BBC CMPW BNEQ BRW CMPW BNEQ INCL	2(R8) 29\$	(R8), 33\$	471
				0B	02	1BF A8 04	31 B1	0020D 00210	29\$:	BRW	56\$ 2(R8), #11	471
				-		04 5B	12	00214 00216		BNEQ	30\$	N_NESTING	472
				ОС	02	7A	11	00218 0021A	30\$:	BRB	5/5), #12	472
				OD		A8 34 5B 8F	12	0021E	300.	BRB CMPW BNEQ SOBGEQ PUSHL CALLS	34\$	N_MESTING, 31\$:
				00	000289FA	8F	F4 DD FB	00223 00229 00230		PUSHL	#166	394 LIB\$SIGNAL	473 473
			00000000	00		56	05	00230 00232	31\$:	TSTL	OPTO	R_INDEX	473
					00000000	EF	9F	00234		PUSHAB	P. AWI	R_INDEX	473
			00000000	00	00028362	8F	DD	00236		PUSHL	#164	706	
			0000000G	56	20	92	CS	0023C 00242 00249	32\$:	TSTL BGTR PUSHAB PUSHL PUSHL CALLS SUBL2	#2,	706 LIB\$SIGNAL OPTOR_INDEX ATOR_STACK+4[OPTOR_INDEX], TOKEN	473
				AE	28 /		DO	0024C 00252 00254 00258 0025A	33\$: 34\$:	MOVL BRB CMPW BNEQ CLRL	39\$	ATOK_STACK+4LOPTOR_INDEXJ, TOKEN	473 473 470 474
				39	02	A8	12	00254	548:	BNEQ	38\$), #5/	:
					04	AE	04	0025A		CLRL	OPER	AND_EXPECTED	: 474

						1	K 10 6-Sep-1 4-Sep-1	984 02:10 984 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 168 (21)
		19		57 57 00 8F 01 00 50 03 AD 47	D6	0025D		INCL	OPAND_INDEX OPAND_INDEX, #25	: 4749 : 4750
			000280E0	OD.	19	00262 00264		BLSS PUSHL	35\$ #164064	:
	900000000	00	000200E0	01	FB	0026A 00271	***	CALLS	#1 I IDECTONAL	4752
	0000V 9C A			50	FB DO	00276	35\$:	MOVL BBC	RO, OPERAND_STACK[OPAND_INDEX]	: 4754
09	0000000G	00	90	AD47	E1 DD	0027B 00283		PIISHI	#0, GET_SET_CONSTANT R0, OPERAND_STACK[OPAND_INDEX] #3, DBG\$GL_DEVELOPER, 36\$ OPERAND_STACK[OPAND_INDEX] #1, DUMP_PRIMARY OPERATOR_STACK[OPTOR_INDEX], TOKEN OPTOR_INDEX 438	4755
	0000V	CF	24	VI	10	00287	36\$:	CALLS	#1, DUMP PRIMARY	:
	10	ME	24	56 7F	D0	00292 00294		DECL	OPTOR_INDEX	: 4759 : 4760
		31	02	A8	B1	00296	37\$: 38\$:	CALLS MOVL DECL BRB CMPW	43\$ 2(R8), #49	: 4706 : 4770
		50	OC	33	81 12 9A	0029A 0029C		BNEQ	40\$ 12(R8), R0	4774
		50 50	04	04	Ć6 9F	002A0		DIVLE	#4, RO 4(RO)	:
	0000000G	00	04	01	FB	00246		CALLS	#1. DBG\$GET_TEMPMEM	4773
		5A	OC	50 A8	00 9A	002AD 002B0 002B4 002B7 002BB		DIVL2 PUSHAB CALLS MOVL MOVZBL	#1, DBG\$GET TEMPMEM RO, TEMP TOKEN 12(R8), RO	: 4776
6A		50 50 68		0D	C0 28	002B4		ADDL2 MOVC3	#13, RO RO, (R8), (TEMP_TOKEN)	4777
-	18	AE	10	5A	DO	002BB		MOVL	TEMP TOKEN, TOKEN	: 4778
	0000V	CF	18	01	DD FB AA	002BF 002C2		CALLS BICW2	#1, GET_FIELDREF #512, aTOKEN	4779
	18	BE	0200	8F 7D	- 11	002CD	398:	BICW2 BRB	46\$: 4780 : 4706
		34	02	A338400108D00AE1FD82000A5	B1 12 FB	002CF	40\$:	BRB CMPW BNEQ	2(R8), #52 41\$: 4788
	0000V	CF		ÓÖ	FB	002D3 002D5		CALLS	#O, OPERATOR_TO_RESTORE_RADIX	: 4790
	000000000	AE EF		OA	D0	002DE		MOVL	RO, TOKEN #10, EXPRESSION_RADIX	4791
		35	02		11 B1	002E5 002E7	415:	BRB CMPW	46\$ 2(R8), #53	: 4706 : 4799
	0000v	CF		12	B1 12 FR	002E7 002EB 002ED		BNEQ	#0. OPERATOR_TO_RESTORE_RADIX	: 4801
	00000000	ĀĒ		50	FB DO DO	002F2		MOVL	RO, TOKEN #16, EXPRESSION_RADIX	:
	0000000			6A	- 11	002F6 002FD 002FF		BRB	403	: 4802 : 4706
		36	02	12	B1 12 FB D0	002FF 00303	428:	BRB CMPW BNEQ CALLS	2(R8), #54 44\$: 4810
	0000V	CF		90	FB	00303 00305 0030A		CALLS	#O, OPERATOR_TO_RESTORE_RADIX	: 4812
	000000000	AE EF		08	DO	0030E	170.	MOVL	#8, EXPRESSION_RADIX	4813
		37	02	A8	81	00315 00317	438:	BRB CMPW BNEQ	#8, EXPRESSION_RADIX 50\$ 2(R8), #55	: 4706 : 4821
	0000V	CF		12	81 12 FB DO	0031B 0031D		CALLS	#0. OPERATOR_TO_RESTORE_RADIX	4823
	00000000	AE		A820500 500 6A8200508 7A820509 A100509 A170509	DO	00322 00326		MOVL	RO, TOKEN	:
			02	59	D0	0032D	150	BRB	50\$	4824
	0041	8F	02		B1 12 D0	0032F 00335	45\$:	BRB CMPW BNEQ	2(R8), #65 47\$	4836
		5A 2C	. 02	AE46 AA 48	D0 B1 12	00337 00330		MOVL CMPW BNEQ	OPERATOR_STACK[OPTOR_INDEX], TEMP_TOKEN 2(TEMP_TOKEN), #44 51\$	4838 4839

				16-Sep-19 14-Sep-19	84 02:10 84 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 169 (21)
18		00000000.	56 D7 003 EF 9E 003 64 11 003 A8 B1 003	4C 46\$:	DECL MOVAB BRB	OPTOR_INDEX COBOL_NOT_EQL_TOKEN, TOKEN 54\$: 4842 : 4843 : 4839
	3F	02	A8 B1 003	4C 46\$: 4E 47\$:	CMPW BNEQ	2(R8), #63 49\$	4839 4853
	5A	24 A	A8 B1 003 17 12 003 E46 D0 003 AA B1 003 28 12 003 56 D7 003	54 59 50	MOVL CMPW BNEQ	OPERATOR_STACK[OPTOR_INDEX], TEMP_TOKEN 2(TEMP_TOKEN), #44 51\$	4855 4856
18	AE	00000000	56 07 003 EF 9E 003 47 11 003	5F 61	MOVAB	OPTOR_INDEX	: 4859 : 4860
0040	8F	02	47 11 003 A8 B1 003	69 48\$:	BRB CMPW BNEQ	COBOL_NOT_GTR_TOKEN, TOKEN 54\$ 2(R8), #64	: 4856 : 4870
	SA		A8 B1 003 24 12 003 E46 D0 003	71	BNEQ	2(R8), #64 52\$ OPERATOR_STACKCOPTOR_INDEX], TEMP_TOKEN	:
	5A 2C	24 A	AA B1 003 0C 12 003	78 70	MOVL CMPW BNEQ	SIS W44	4872 4873
18	AE	00000000	56 D7 003 EF 9E 003 28 11 003 A8 9F 003	80	DECL MOVAB BRB	OPTOR_INDEX COBOL_NOT_LSS_TOKEN, TOKEN 54\$; 4876 ; 4877 ; 4873
		00	A8 9F 003	8A 51\$:	PUSHAB	12(R8)	: 4881
		00028982	01 DD 003 8F DD 003 14 11 003	8F	PUSHL	#1 #166322	
	20	02	14 11 003 A8 B1 003 15 13 003	97 528:	BRB CMPW	53\$ 2(R8), #44	: 4888
		00000000.	A8 B1 003 15 13 003 EF 9F 003 01 DD 003 8F DD 003 03 FB 003 56 D6 003 56 D1 003	9B 9D	BEQL PUSHAB	54\$ P.AWX	: 4899
		00028362	01 DD 003 8F DD 003	A3	PUSHL	#1 #164706	
0000000G	00		03 FB 003 56 D6 003	AB 53\$:	CALLS	#3, LIB\$SIGNAL OPTOR_INDEX	4909
	19		56 D1 003	B4	CMPL	OPTOR_INDEX, #25	: 4910
		000280E0	0D 19 003 8F DD 003 01 FB 003	B9	BLSS PUSHL	55\$ #164064	
000000006	00 AE46	18	AE NO NOZ	r4 EEE.	MOVL	#1, LIB\$SIGNAL TOKEN, OPERATOR_STACK[OPTOR_INDEX]	4911
		F	AE DO 003 C71 31 003 57 D5 003 15 15 003	CC CF 56\$:	BRW TSTL	OPAND_INDEX	4911 4477 4921
		00000001	57 D5 003 15 15 003	01	BLEQ PUSHAB	57\$	1721
		00000000	71 31 003 57 D5 003 15 D5 003 15 DD 003 8F DD	09	PUSHL	P.AWY	
000000006	00 50	00028362	03 FB 003	E1	PUSHL	#164706 #3, LIB\$SIGNAL	
	50 18	9C 04	AD DO 003 AC E8 003	E8 57\$:	MOVL BLBS BLBS CLRL	#3, LIB\$SIGNAL OPERAND_STACK, VALPTR ADDRESS_EXPRESSION, 59\$ DEPOSIT_FLAG, 58\$	4922 4929 4930 4932
	11		AC E8 003 6E E8 003 7E D4 003 50 DD 003	FÕ	BLBS	DEPOSIT_FLAG, 58\$ -(SP)	4930
		00000001	50 DD 003	5	PIICHI	VALPTR	. 4732
00000000	00	00000000	EF 9F 003 03 FB 003	FD	CALLS	DBG\$GL_IDENTITY_TOKEN #3. DBG\$EVAL_LANG_OPERATOR ADDRESS_EXPRESSION, 60\$	
	11	04	7E D4 004	04 58\$: 08 59\$:	PUSHAB CALLS BLBC CLRL PUSHL	-(SP)	: 4939 : 4941
		00000000	50 DD 004 EF 9F 004	OA OC	PUSHL	VALPTR DBG\$GL_IDENTITY_TOKEN	
00000000	00		03 FB 004 04 004	12 19 60\$:	CALLS	#3. DBG\$EVAL_ADDR_OPERATOR	4945
			04 004	., 500.			. 4742

[;] Routine Size: 1050 bytes, Routine Base: DBG\$CODE + 05F8

DBGPARSER V04-000

M 10 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1

Page 170 (21)

```
DBGPARSER
V04-000
                                                                                                                                                                                                                                                                                                                                   VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                        Page 172
(22)
                                                           5003
500067
500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
5500067
550
                                                                                                                     ARG_PTR[.BIF_INDEX] = DBG$EXPRESSION_PARSER(FALSE, COMPAREN_TERM_TBL);
      4891
4892
4893
4894
4896
4896
4896
4890
4901
4903
4904
4907
4908
4909
4911
4912
                                                                                                                           If an invalid terminator was found signal the error. If not, increment the character pointer past the terminator.
                                                                                                                               .TERMINATOR_CODE EQL TOKEN$K_TERM_NONE
                                                                                                                    SIGNAL (DBG$ MISCLOSUB);
CHARPTR = .CHARPTR + .TERMINATOR_LENGTH;
BIF_INDEX = .BIF_INDEX + 1;
                                                                                                                     END:
                                                                                                      BIF_INDEX = .BIF_INDEX - 1;
                                                                                                             If the number of arguments found is not the same as the number of
                                                                                                             expected, signal a bad argument list.
                                                                                                        IF .BIF_INDEX NEQ .LENGTH
                                                                                                                    SIGNAL (DBG$_INVARGLIS, 1, .NAME);
      4913
                                                                                                             If only one argument is requested, a zero is left for the second argument. This is done for the subsequent call to DBG$EVAL_LANG_OPERATOR.
      4915
4916
                                                                                                              Then store away the number of arguments and return a vector of these
                                                                                                              arguments.
      4917
4918
4919
                                                                                                      ARG_PTR[0] = .BIF_INDEX;
      4920
                                                                                                      RETURN .ARG_PTR;
                                                                                                      END:
                                                                                                                                                                                                                                                                                                            DBG$GET BIF_ARGUMENTS, Save R2,R3,R4,R5
LIB$SIGNAL, R5
TERMINATOR_CODE, R4
#1, LENGTH, R0
                                                                                                                                                                                                                          00000
                                                                                                                                                                                                          003C
9E
C1
DD
D1
18
D0
FB
                                                                                                                                                                                                                                                                                 .ENTRY
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      4946
                                                                                                                                                          00000000
                                                                                                                                                                                                                                                                                MOVAB
                                                                                                                                                                                                                             00009
                                                                                                                                                                                                                                                                               MOVAB
                                                                                                                                                                                                                          00010
00015
00017
0001A
0001C
0001F
1$:
                                                                                           50
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      4982
                                                                                                                                                                                                                                                                                ADDL3
                                                                                                                                                                                                                                                                               PUSHL
                                                                                                                                               03
                                                                                                                                                                                                                                                                                                              (SP), #3
                                                                                                                                                                                                                                                                                CMPL
                                                                                                                                                                                                                                                                               BGEQ
                                                                                                                                                                                                                                                                                MOVL
                                                                                                                                                                                                                                                                                                            #1, DBG$GET_TEMPMEM
RO, ARG_PTR
#1, BIF_INDEX
TERMINATOR_CODE
TERMINATOR_CODE, #2
                                                                                                                                               00
53
52
                                                                                                      0000000G
                                                                                                                                                                                                                                                                               CALLS
                                                                                                                                                                                                                                                                                MOVL
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      4988
4989
4990
                                                                                                                                                                                                                                                                               MOVL
                                                                                                                                                                                                                                                                               CLRL
CMPL
BEQL
                                                                                                                                               02
                                                                                                                                                                                                                                                                                                            BIF_INDEX, LENGTH
                                                                                                                             04
                                                                                                                                                                                                                                                                               CMPL
BLEQ
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      4996
                                                                                                                                                                                                                            00037
00039
                                                                                                                                                                                                                                                                               PUSHL
                                                                                                                                                                                                                                                                                                             NAME
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      4998
                                                                                                                                                                                                                 DD
                                                                                                                                                                                                                  DD
                                                                                                                                                                                                                                                                                PUSHL
                                                                                                                                                                                                                 DD
FB
9F
                                                                                                                                                                                                                                                                                                             #165944
                                                                                                                                                                                                                                                                               PUSHL
                                                                                                                                                                                                                                                                                                            #3, LIB$SIGNAL COMPAREN_TERM_TBL
                                                                                                                                                                                                                                                                               CALLS
PUSHAB
```

00000000

DE

DBGPARSER V04-000		C 11 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 173 (22)
	FB92 6342 65 00028E90 FBCC C4 04 04 AC 08 00028838	7E D4 0004D CLRL -(SP) 02 FB 0004F CALLS #2, DBG\$EXPRESSION_PARSER 50 D0 00054 MOVL RO, (ARG_PTR)[BIF_INDEX] 64 D5 00058 TSTL TERMINATOR_CODE 09 12 0005A BNEQ 4\$ 8F DD 0005C PUSHL #167568 01 FB 00062 CALLS #1, LIB\$SIGNAL A4 C0 00065 4\$: ADDL2 TERMINATOR_LENGTH, CHARPTR 32 D6 0006B INCL BIF_INDEX BF 11 0006D BRB 2\$ 52 D7 0006F 5\$: DECL BIF_INDEX, LENGTH 0E 13 00075 BEQL 6\$	5008 5010 5011 5012 4990 5016 5021 5023
	65 63 50	AC DD 00077 PUSHL NAME 01 DD 0007A PUSHL #1 8F DD 0007C PUSHL #165944 03 FB 00082 CALLS #3, LIB\$SIGNAL 52 DO 00085 6\$: MOVL BIF_INDEX, (ARG_PTR) 53 DO 00088 MOVL ARG_PTR, R0 04 0008B RET	5030 5032 5033

; Routine Size: 140 bytes, Routine Base: DBG\$CODE + 0A12

.....

............

GLOBAL ROUTINE DBG\$LEXICAL_SCANNER(CPERAND_EXPECTED, ADDRESS_EXPRESSION, TERM_LIST, PAREN_NESTING) =

FUNCTION

This routine is the Lexical Scanner used during expression parsing. It scans the character or characters pointed to by CHARPTR to pick up the next lexical token according to the rules of the current language. It picks up or constructs a Lexical Token Entry for the found lexical token and returns a pointer to that Token Entry as its result.

The Lexical Scanner is called by the Primary Parser which then uses the returned token to build up a Primary Descriptor if the token is part of a Primary Symbol. If the returned token is not part of a Primary Symbol (i.e., if it is an operator in the current language or the current Address Expression or if it is a constant), the Primary Parser returns it directly to the Expression Parser. The Expression Parser is thus fed a stream of operands and operators which it then uses to interpret and evaluate the current expression.

This routine assumes that the input line being scanned has already been converted to upper case and it assumes that the input line is terminated by a carriage-return character. It also assumes that the variable CHARPTR has been set up to point to the current position in the buffer being scanned. CHARPTR is updated by this routine to point to the first character position after the current token.

This routine accepts a list of allowed "terminator tokens" (keywords such as "DO" or "THEN" or special characters such as ",", ")", or "=", depending on context). This list is passed to the Lexical Scanner which returns the Terminator Operator when such a token or a carriage-return is encountered. As a side effect, OWN variable TERMINATOR CODE is set to a value which indicates which terminator token was found. That terminator's character length is also set in TERMINATOR LENGTH. (This side effect is used when parsing subscript expressions.)

INPUTS

OPERAND_EXPECTED - A flag set to TRUE if an operand is expected next in the parse of the current expression. This flag is used to disambiguate certain operators, such as "+" and "-", which are prefix operators when an operand is expected next and are infix operators when an operator is expected next.

ADDRESS_EXPRESSION - A flag set to TRUE if we are parsing a DEBUG Address Expression instead of a language expression. This affects the parsing of Address Expression operators such as '+', '-', '*', ',', and 'a' which are recognized by DEBUG rules, not language rules, in Address Expressions.

TERM_LIST - A vector of pointers to Terminator Lexical Token Entries for the Terminator Tokens which can terminate the expression to be parsed. The vector must be in PLIT form (TERM_LIST[-1] gives the number of entries) and each pointer is expected to be relative to TABLEBASE. If there are no terminator tokens other than carriage return, this list is empty (0 entries).

PAREN_NESTING - The current parenthesis nesting depth. This parameter is used to detect whether certain tokens are expression terminators or not. (for example, a ")" token in a subscript expression terminates it only if parentheses are already balanced.)

A pointer to the Lexical Token Entry for the next lexical token found in the input line is returned as the routine value. If there are no more lexical tokens on the line, a pointer to a Token Entry for the TOKENSK_TERMINATOR operator is returned.

BEGIN

MAP

TERM_LIST: REF VECTOR[,LONG]; ! Pointer to Terminator Table to use

CHECK_THIS_TERMINATOR;

Label used to leave terminator checking code

ACTION, BACKUP_DIGIT_PTR, BACKUP_NUMBER_KIND, BEST_TOKEN_FOUND,

CLASS.

ENDPTR, ERRORMSG, INDEX, NAMEPTR: REF VECTOR[,BYTE], NEW STARTPTR, NUMBER KIND, PRID: REF PRIDSENTRY, QUOTE,

STATE_INDEX,

TERMPTR: REF TOKENSENTRY,
TOKEN: REF TOKENSENTRY,
TOKENBUFFER: VECTOR[256,BYTE],

TOKEN TYPE,

Action index during number scanning
Pointer to last good digit in number
--used to back up number scan
Kind of numeric constant definite so
far--used to back up number scan
Pointer to the Operator Token Entry
for an operator with the right
name but the wrong kind
Character class code of current character during number scanning
Pointer to last char in an identifier
Error message condition code
Index into look-up tables
Pointer to name string in Percent Tbl
Start pointer to lower case identifier
Kind of Numeric Constant Token found
Pointer to Predefined Identifier Entry
Quote character which started the
current quoted string constant
Pointer to Start of current token
Current Number Scanner State Table
index during number scanning
Pointer to Terminator Toen Entry
Pointer to Operator Token Entry
Vector in which current token is
accumulated as Counted ASCII
Token's type; eg. TOKEN\$K_STRING
The character length of current token

! Start by scanning past any leading blanks. Then mark the start location ! of the token to be picked up.

so C is special-cased below).

"a" is an operator in some languages. The only one where it is

a prefix operator (the case that causes ambiguities) is PLI, where it means "not". We resolve this by simply

Page 178 (23)

D

VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1

```
assuming that """ means previor if this is an address expression, and means "not" in a language expression. This essentially disallows "" for previor in language expressions (%PREVLOC can
be used instead).
                                                           "." is highly overloaded. In many languages it can be the start of a floating point number, so we check for the next character being a digit here. It can also be the indirection operator in an address expression. We check for the next character being "(", ", or "\", and if so, assume that the dot means indirection and not current location.
                                                             (NOT .CHARTBL[.CHARPTR[0], CHRTBL$V_IDENT_START]) AND (NOT .CHARTBL[.CHARPTR[0], CHRTBL$V_DIGIT]) AND (.CHARPTR[0] NEQ '%' OR
                                                               (.DBG$GB_LANGUAGE EQL DBG$K_C AND (NOT .ADDRESS_EXPRESSION)))
                                                         THEN
                                                                BEGIN
                                                                IF .STARTPTR[0] EQL '\' THEN RETURN CURVAL_TOKEN; IF (.STARTPTR[0] EQL 'A') AND
                                                                    ((.DBG$GB_LANGUAGE NEQ DBG$K_PLI) OR .ADDRESS_EXPRESSION)
                                                                THEN
                                                                     RETURN PREVLOC TOKEN;
.STARTPTR[0] EQL '.' AND
NOT ((.CHARPTR[0] EQL '(') OR
(.CHARPTR[0] EQL '.') OR
(.CHARPTR[0] EQL '\'))
                                                                THEN
                                                                       RETURN CURLOC_TOKEN:
                                                                END:
                                                         CHARPTR = .STARTPTR;
                                                        END:
                                                     If we are expecting an operand next, check for the special DEBUG symbols that begin with a percent sign "%". This includes %LINE, %LABEL, %NAME,
                                                     and all the register names.
                                                 IF (.CHARPTR[O] EQL '%') AND .OPERAND_EXPECTED AND
                                                 THEN THEN THE THE THE THE THE THEN
                                                        BEGIN
                                                            Accumulate the identifier after the "%"-sign.
                                                         CHARPTR = .CHARPTR + 1;
                                                         TOKENLEN = 1
                                                         TOKENBUFFER[1] = '%';
                                                         WHILE .CHARTBL[.CHARPTR[0], CHRTBL$V_ALPHABETIC] OR .CHARTBL[.CHARPTR[0], CHRTBL$V_DIGIT]
                                                         DO
                                                                 TOKENLEN = . TOKENLEN + 1;
```

```
DBGPARSER
V04-000
                                                                                                                                                                                                           16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                                                                                                                                       VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                                                                                                  TOKENBUFFER[.TOKENLEN] = .CHARPTR[0];
    $\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 678\\ 67
                                                                                                                  CHARPTR = . CHARPTR + 1:
                                                                                                                  END:
                                                                                                     TOKENBUFFER[0] = .TOKENLEN:
                                                                                                           Now look up the "%"-symbol in the DEBUG Percent Table.
                                                                                                     INDEX = PERCENT_NOFIND;
INCR I FROM 0 TO .PERCENT_TABLE[-1] - 1 DO
BEGIN
                                                                                                                  NAMEPTR = .PERCENT_TABLE[.1] + TABLEBASE;
IF CHSEQL(.NAMEPTR[1], NAMEPTR[2], .TOKENLEN, TOKENBUFFER[1], 0)
                                                                                                                              BEGIN
INDEX = .NAMEPTR[0];
                                                                                                                               EXITLOOP;
                                                                                                                               END:
                                                                                                                 END:
                                                                                                          Now do whatever further processing is appropriate for this "%"-symbol.
                                                                                                    CASE .INDEX FROM PERCENT_NOFIND TO PERCENT_IDENT OF SET
                                                                                                                       Handle the No-find case. We do not recognize this "%"-symbol,
                                                                                                                       so we do nothing.
                                                                                                                  [PERCENT_NOFIND]:
                                                                                                                       Handle %LINE and %LABEL. Here we pick up the line or label
                                                                                                                        number that follows the keyword and construct an Identifier
                                                                                                                        Token Entry for the line or label and return that to the caller.
                                                                                                                 PERCENT_LINE,
PERCENT_LABELJ:
BEGIN
                                                                                                                                    Set up the fully spelled out keyword (%LINE or %LABEL) in
                                                                                                                                    TOKENBUFFER and set up the appropriate error message code.
                                                                                                                               IF .INDEX EQL PERCENT_LINE THEN
                                                                                                                                           BEGIN
                                                                                                                                            CH$MOVE(6, UPLIT BYTE(%ASCII '%LINE '), TOKENBUFFER[1]);
                                                                                                                                            TOKENLEN = 6
                                                                                                                                            ERRORMSG = DBG$_SYNERRLINE;
                                                                                                                              ELSE
```

Page 181 (23)

Page 183 (23) 5660

```
[PERCE BIN]:
RETURN RADIX_OP_BIN;

Any other CASE index i
```

! Any other CASE index is an internal DEBUG error. [INRANGE, OUTRANGE]:

\$DBG_ERROR('DBGPARSER\LEXICAL_SCANNER 20');

TES:

This is the end of the '%'-symbol processing. If we have not recognized the symbol yet, it is not a DEBUG special symbol so we reset CHARPTR to point to the '%' sign again. The hope is that we will recognize it as a valid token later in the Lexical Scanner.

CHARPTR = .STARTPTR;

END:

! End of '%'-symbol scanning

See if this token is a quoted character string. Quoted character strings in this context are defined to start with a quote character (usually "or "), to continue as a string of zero or more non-quote characters, and to be terminated by a quote character. The starting and ending quote characters must be the same character and that quote character is represented within the string by two consecutive quote characters. If we find such a quoted string, we accumulate it and return a String Constant Token.

IF .CHARTBL[.CHARPTR[0], CHRTBL\$V_STRING_QUOTE]
THEN

BEGIN
SCAN_QUOTED_STRING(TOKENBUFFER, TOKEN_TYPE);
RETURN CREATE_OPERAND_TOKEN(.TOKEN_TYPE, TOKENBUFFER);
END;

See if this token is a numeric constant. If so, we pick up the character representation of the number and return it as a Numeric Constant Lexical Token Entry. This scan must be done before the Identifier scan below so that COBOL integers get interpreted as numbers, not identifiers.

This code simulates a finite-State Machine (FSM) which accepts any valid numeric constant in the current language. The machine is defined by a state table where each state has a set of allowed transitions to other states. Each transition is selected by the next input character and has an associated action routine defined below. When a numeric constant has been recognized, a transition is taken whose action routine builds and returns a Lexical Token Entry for the numeric constant.

IF .CHARTBL[.CHARPTR[O], CHRTBL\$V_NUMBER_START] AND ((NOT .ADDRESS_EXPRESSION) OR (.CHARPTR[O] NEQ '.'))
THEN
BEGIN

Page 185 (23)

DI

V

Loop through the transitions from this state until we find a transition for this character class or for NUMSTSK_CLASS_OTHER (the class of all other characters). Pick up the action index and next state associated with this transition of the finite-

WHILE (.STATE_TABLE[.STATE_INDEX, NUMST\$B_CHAR_CLASS] NEQ

V

VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1

Some state transitions require no semantic action, so we

Go past a digit in the integer part of the number. Set the digit backup pointer to the last good digit seen (this one).

At the decimal point, mark that we have a floating-point

At a fraction digit, set the backup pointer to the last good digit seen (i.e., this one) and note that we really have a floating-point number even if we must back up the scan.

Mark that we found a D exponent marker. [NUMST\$K_ACT_MARK_D_EXP]:

Page 187 (23)

```
DBGPARSER
V04-000
                                                                                                                                                      VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1
                                                                            IF .EXPRESSION_RADIX NEQ DBG$K_HEX
   BEGIN
CHARPTR = .STARTPTR;
                                                                                  EXITLOOP;
                           END:
                                                                           END:
                                                                       In COBOL, a number such as 123 can be a name. Such a name must be entered as %NAME 123. However, if we find a number followed by a valid identifier character, as in 123A or 12-3, then we actually have an identifier, so we exit the number scanning loop without returning a number token. If the
                                                                       number ended with any other character, we return a valid
                                                                       number token.
                                                                     [NUMST$K_ACT_COB_CKNUM]:
                                                                           IF .CHARTBL[.CHARPTR[0], CHRTBL$V_IDENT_START] OR .CHARTBL[.CHARPTR[0], CHRTBL$V_IDENT_MIDDLE] OR .CHARTBL[.CHARPTR[0], CHRTBL$V_IDENT_END]
                                                                                  BEGIN
CHARPTR = .STARTPTR;
                                                                                  EXITLOOP:
                                                                           IF NOT ((.NUMBER_KIND EQL TOKEN$K_HEX_INTEGER) OR (.NUMBER_KIND EQL TOKEN$K_OCT_INTEGER) OR
                                                                                         (.NUMBER_KIND EQL TOKEN$K_BIN_INTEGER))
                                                                           THEN
                                                                                  NUMBER_KIND = TOKEN$K_PACK_DECIMAL;
                                                                          TOKENLEN = .CHARPTR - .STARTPTR;
IF .TOKENLEN GTR 255 THEN SIGNAL(DBG$_NUMCONLONG);
TOKENBUFFER[0] = .TOKENLEN;
CH$MOVE(.TOKENLEN, .STARTPTR, TOKENBUFFER[1]);
RETURN CREATE_OPERAND_TOKEN(.NUMBER_KIND, TOKENBUFFER);
                                                                           END:
                                                                       Save the base of a number in ADA, where a number can be
                                                                       of the form base#number.
                                                                    [NUMST$K_ACT_SAVE_BASE]:
                                                                       Any other action index constitutes an internal error.
                                                                    INMSTSK_ACT_GIVE_ERROR,
                                                                      OUTRANGE ]:
                                                                           $DBG_ERROR('DBGPARSER\LEXICAL_SCANNER 30');
```

D

Page 189 (23)

```
DBGPARSER
V04-000
                                                                                                                                             VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGPARSER.B32:1
                                                                TES:
  Go on to the next character in the buffer and loop.
                                                          CHARPTR = .CHARPTR + 1:
                                                                                                      ! End of WHILE loop over number states
                                                         END:
                                                   END:
                                                                                                      ! End of numeric constant scanning
                                               See if this token is an identifier according to the rules of the current language. If so, pick up the whole identifier, see if this identifier is actually an operator (such as AND or MOD) in the current language, and
                                                return either an Identifier Token or an Operator Token.
                                             IF .CHARTBL[.CHARPTR[O], CHRTBL$V_IDENT_START]
                                             THEN
                                                   BEGIN
                                                      We have a valid start character for an identifier in the current language. Now scan through the identifier as long as we have valid
                                                      middle characters and set ENDPTR each time we find a valid end character for an identifier. At the end of the scan we set CHARPTR to point to the first character after the identifier end character.
                                                   ENDPTR = .CHARPTR - 1;
WHILE TRUE DO
                                                          BEGIN
                                                          IF .CHARTBL[.CHARPTR[0], CHRTBL$V_IDENT_END]
                                                          THEN
                                                                ENDPTR = .CHARPTR;
IF NOT .CHARTBL[.CHARPTR[O], CHRTBL$V_IDENT_MIDDLE] THEN EXITLOOP;
                                                         CHARPTR = .CHARPTR + 1;
IF (NOT .CHARTBL[.CHARPTR[0], CHRTBL$V_IDENT_MIDDLE]) AND (NOT .CHARTBL[.CHARPTR[0], CHRTBL$V_IDENT_END])
                                                               THEN EXITLOOP:
                                                         END:
                                                   CHARPTR = .ENDPTR + 1;
                                                      Copy the identifier name to TOKENBUFFER.
                                                   TOKENLEN = .CHARPTR - .STARTPTR;

IF .TOKENLEN GTR 255 THEN SIGNAL(DBG$ IDENTLONG);

CH$MOVE(.TOKENLEN, .STARTPTR, TOKENBUFFER[1]);

TOKENBUFFER[0] = .TOKENLEN;
                          6001
                                                    ! If the language is C, then we want to make sure that the identifier
                                                    ! preserves the original casing (upper/lower). This is because
```

Page 190 (23)

```
H 12
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                             VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
DBGPARSER
V04-000
                                                     in C, upper case XXX is a different variable from lower case xxx, for example. So we copy characters from the original command buffer instead of the up-cased command buffer.

We use a flag "CASING_SIGNIFICANT" which is set in the SET_LANGUAGE routine for this. At present, C is the only language that sets this flag to TRUE.
   . CASING_SIGNIFICANT
                                                    THEN
                                                         BEGIN
IF (.
                                                              (.STARTPTR LSS .DBG$GL_UPCASE_COMMAND_PTR[0]) OR (.STARTPTR GTR .DBG$GL_UPCASE_COMMAND_PTR[1])
                                                                $DBG_ERROR('DBGPARSER\DBG$LEXICAL_SCANNER 40');
                                                         END:
                                                      See if this identifier is actually the name of an operator. We scan a language-specific operator table to determine this. If so, return
                                                      the corresponding Operator Token.
                                                   BEST_TOKEN_FOUND = 0;
INCR_I FROM 0 TO .IDENT_OPERATOR_TABLE[-1] - 1 DO
                                                         TOKEN = .IDENT OPERATOR TABLE[.I] + TABLEBASE;
IF CHSEQL(.TOKENETOKENSB_OPLEN],
TOKENETOKENSA_OPNAME], .TOKENLEN, TOKENBUFFER[1], 0)
                                                              RETURN . TOKEN
                                                                ELSE
                                                                      BEST_TOKEN_FOUND = .TOKEN;
                                                                END:
                                                         END:
                                                   IF .BEST_TOKEN_FOUND NEG O THEN RETURN .BEST_TOKEN_FOUND;
                                                      See if this identifier is the name of a built-in function. We scan a language-specific built-in function table to determine this. If
                                                      so, return the corresponding Operand Token.
                                                   BEST_TOKEN FOUND = 0;
INCR I FROM 0 TO .BIF_TABLE[-1] - 1 DO
BEGIN
```

Page 191 (23)

Page 192 (23)

```
CHSMOVE (.TOKENLEN, .STARTPTR, TOKENBUFFER[1]):
                       6118
61120
61121
61121
611223
611223
611223
611223
611233
611333
61133
61133
61141
                                                     We now have an operator symbol in TOKENBUFFER. Look it up in the operator table for the current language, and if found, return the corresponding Operator Token. Note that we only accept an operator if it is a Primary operator (which we accept in both language and
                                                     address expressions) or if we are in a language expression.
                                                 BEST_TOKEN FOUND = 0;
INCR I FROM 0 TO .OPCHAR_OPERATOR_TABLE[-1] - 1 DO
BEGIN
TOKEN = .OPCHAR_OPERATOR_TABLE[.I] + TABLEBASE;
IF (.TOKEN[TOKENSY_PRIMARY] OR (NOT .ADDRESS_EXPRESSION)) AND
CHSEQL(.TOKEN[TOKENSB_OPLEN],
TOKEN[TOKENSB_OPLEN],
                                                                            TOKEN[TOKENSA_OPNAME], .TOKENLEN, TOKENBUFFER[1], 0)
                                                        THEN
                                                               BEGIN
                                                               IF (.OPERAND_EXPECTED AND

(.TOREN[TOKEN$B_KIND] EQL TOKEN$K_PREFIX_OP)) OR

((NOT .OPERAND_EXPECTED) AND
                                                                            (.TOKEN[TOKEN$B_KIND] NEQ TOKEN$K_PREFIX_OP))
                                                               THEN
                                                                     RETURN . TOKEN
                       6143
6144
6144
6146
6147
6148
6151
6157
6157
6157
6161
6161
                                                               ELSE
                                                                     BEST_TOKEN_FOUND = .TOKEN;
                                                               END:
                                                        END:
                                                    If we found an operator but it is not in a valid context (e.g. infix
                                                     operator when we expect a prefix operator), we return it anyway but
                                                     only if it cannot be an address expression operator (which might be
                                                     valid in the current context).
                                                 IF (.BEST_TOKEN_FOUND NEQ 0) AND
NOT (.ADDRESS_EXPRESSION AND
.CHARTBLE.STARTPTR[0], CHRTBL$V_ADDRESS_OP])
                                                        RETURN .BEST_TOKEN_FOUND;
                                                     There is no such operator. Reset CHARPIR to point to the start of
                                                     the token to give the code below a chance to make sense of it.
                                                  CHARPTR = .STARTPTR;
                       6166
6167
                                                  END:
                                                                                                       ! End of operator scanning
                       6168
                       6169
6170
6171
6172
6173
```

If we are parsing an Address Expression at present, we check for the specific operators allowed in Address Expressions. We do not use the anguage-specific rules for combining operator characters in this case. If we find such an operator here, return the corresponding Token Entry.

Page 194 (23)

```
DBGPARSER
V04-000
                                                                                                                                VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
 END:
                                                       Return "." as the FORTRAN record component selection operator.
                                                     IF (NOT .OPERAND_EXPECTED) AND (.STARTPTR[0] EQL '.')
                                                    THEN
                                                         BEGIN
CHARPTR = .STARTPTR + 1;
RETURN FORTRAN_DOT_TOKEN;
                                                    END:
                                                 Handle C. Here we pick up the C operators prefix &, infix &, infix &, prefix and postfix ++, prefix and postfix --, +, -, and ->. Possible ambiguities involving these operators must be
                                                 resolved using C rules, which is what we do here.
                                              EDBG$K C]:
                                                       Check for the operators that begin with '&': address-of,
                                                       bit-and, and short-and.
                                                     IF . CHARPTR[O] EQL '&'
                                                    THEN
                                                          BEGIN
                                                          CHARPTR = .CHARPTR + 1;
IF .CHARPTR[0] EQL '&'
                                                          THEN
                                                                CHARPTR = .CHARPTR + 1;
                                                                RETURN C_AND_TOKEN;
                                                          IF .OPERAND_EXPECTED THEN RETURN C_ADDR_OF_TOKEN; RETURN C_BIT_AND_TOKEN;
                                                          END:
                                                       Check for the operators that start with '+': add, pre-increment, and post-increment. (X+Y ++X X++)
                                                     IF .CHARPTR[0] EQL '+'
                                                           CHARPTR = .CHARPTR + 1;
If .CHARPTR[0] EQL '+'
                                                           THEN
                                                                BEGIN
CHARPTR = .CHARPTR + 1;
IF .OPERAND_EXPECTED
THEN
```

```
B 13
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
  RETURN C_MINUS_TOKEN
                                                          ELSE
                                                                RETURN C_SUB_TOKEN;
                                                          END:
                                                    END:
                                                 Handle RPG. The special indicator names start with "*".
                                               [DBG$K RPG]:
                                                    BEGIN
IF .CHARPTR[0] EQL '*'
THEN
                                                          BEGIN
                                                             Pick it up as a name.
                                                          IF .OPERAND_EXPECTED THEN
                               5555666666666666777
                                                                BEGIN
                                                                CHARPTR = .CHARPTR + 1;
                                                                    .CHARTBLE.CHARPTREOJ, CHRTBLSV_IDENT_START]
                                                                      BEGIN
                                                                         We have a valid start character for an identifier in the current
                                                                        language. Now scan through the identifier as long as we have valid middle characters and set ENDPTR each time we find a valid end character for an identifier. At the end of the scan we set CHARPTR to point to the first character after the identifier end character.
                                                                      ENDPTR = .CHARPTR - 2;
WHILE TRUE DO
                                                                            BEGIN
                                                                            IF .CHARTBL[.CHARPTR[0], CHRTBL$V_IDENT_END]
                                                                           THEN
                                                                                 BEGIN
                                                                                 ENDPTR = .CHARPTR;
                                                                                  IF NOT . CHARTBLE. CHARPTREOJ, CHRTBLSV_IDENT_MIDDLEJ THEN EXITLOOP;
                                                                            CHARPTR = .CHARPTR + 1;
                                                                                (NOT .CHARTBLE.CHARPTREO], CHRTBL$V_IDENT_MIDDLE]) AND (NOT .CHARTBLE.CHARPTREO], CHRTBL$V_IDENT_END])
                                                                            THEN EXITLOOP:
                               6666666666
                                                                            END:
                                                                      CHARPTR = .ENDPTR + 1;
                                                                        Copy the identifier name to TOKENBUFFER.
                                                                      TOKENLEN = .CHARPTR - .STARTPTR;
IF .TOKENLEN GTR 255 THEN SIGNAL (DBG$ IDENTLONG);
                                                                      CH$MOVE(.TOKENLEN, .STARTPTR, TOKENBUFFER[1]);
```

DI

```
C 13
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                             VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
                                                                    TOKENBUFFER[0] = .TOKENLEN;
RETURN CREATE_OPERAND_TOKEN(TOKEN$K_IDENTIFIER, TOKENBUFFER);
  END
                                                           Pick it up as an operator.
                                                         ELSE
                                                              BEGIN
                      CHARPTR = .CHARPTR + 1;
                                                              RETURN RPG_MULTIPLY_TOKEN;
                                                         END:
                                                   END:
                                               Handle Ada. Here we distinguish the tick operator from the single quote-both are represented by the character "". If it is a tick,
                                                we return the Tick Operator Token and if it is a single quote, we
                                                pick up the character it quotes and return a Character Constant Token.
                                             [DBG$K ADA]:
                                                   IF .CHARPTR[0] EQL """
                                                   THEN
                                                        BEGIN
                                                           If we are expecting an infix or postfix operator, this must be the "tick" character, which begins one of the postfix tick operators ("'FIRST", "LAST", ...).
                                                         IF NOT . OPERAND_EXPECTED
                                                         THEN
                                                              BEGIN
                                                                 Make a copy of the tick token.
                                                              TOKEN = DBG$GET_TEMPMEM (TOKEN$K_FIXED_SIZE_LONG + 4);
CH$MOVE (TOKEN$K_FIXED_SIZE_BYTE+1,ADA_TICK_TOKEN,.TOKEN);
                                                                Look up which tick operator this is.
                                                              INCR INDEX FROM TOKENSK_TICK_MIN TO TOKENSK_TICK_MAX DO
                                                                    NAMEPTR = .ADA_TICK_TABLE[.INDEX] + TABLEBASE;
                                                                    IF CHSEQL (.NAMEPTR[0], NAMEPTR[1], .NAMEPTR[0], CHARPTR[1])
                                                                    THEN
                                                                         CHARPTR = .CHARPTR + 1 + .NAMEPTR[0];
TOKEN[TOKEN$W_SUBCODE] = .INDEX;
TOKEN[TOKEN$B_OPLEN] = .NAMEPTR[0] + 1;
```

Page 199 (23) D

```
DBGPARSER
V04-000
                                                                                                                 16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                           VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
                                                                                                                                                                                                                          Page 200
(23)
                                                                                           CHSCOPY(1, CHSPTR(UPLIT(''')), .NAMEPTR[0], NAMEPTR[1], %C'', .NAMEPTR[0]+1, TOKEN[TOKENSA_OPNAME]);
  Look for a left paren that would indicate a list of arguments follow the tick operator.
                                                                                               If a paren is found, the subcode value is set set to the corresponding tick operator type by adding 1 to it. Also, the CHARPTR will be
                                                                                               updated to pointer at the character just past
                                                                                               the paren.
                                                                                           WHILE .CHARTBL[.CHARPTR[0], CHRTBL$V_SPACE] DO CHARPTR = .CHARPTR + 1;
                                                                                                 .CHARPTR[0] EQL '('
                                                                                            THEN
                                                                                                  BEGIN
TOKEN[TOKEN$V_ARGUMENT_LIST] = TRUE;
CHARPTR = .CHARPTR + 1;
                                                                                                   TOKEN[TOKEN$V_ARGUMENT_LIST] = FALSE;
                                                                                            RETURN . TOKEN:
                                                                                            END:
                                                                                    END:
                                                                                If we fall through to here, we failed to find
                                                                                a matching tick operator in our table.
                                                                             TOKENBUFFER[0] = 1; ...
                                                                            CHARPTR = .CHARPTR + 1;
WHILE .CHARTBL[.CHARPTR[0], CHRTBL$V_IDENT_START] OR
.CHARTBL[.CHARPTR[0], CHRTBL$V_IDENT_MIDDLE] OR
.CHARTBL[.CHARPTR[0], CHRTBL$V_IDENT_END] DO
                                                                                    BEGIN
                                                                                    IF (.CHARPTR[0] EQL CAR_RET) OR (.TOKENBUFFER[0] GEQ 32)
THEN EXITLOOP;
TOKENBUFFER[0] = .TOKENBUFFER[0] + 1;
TOKENBUFFER[.TOKENBUFFER[0]] = .CHARPTR[0];
CHARPTR = .CHARPTR + 1;
                                                                             SIGNAL (DBG$_UNKATTRIB, 1, TOKENBUFFER);
                                                                         Otherwise we are expecting an operand, so it must be the single quote character. Pick up the single character quoted and return a Character Constant Lexical Token Entry.
                                                                       IF (.CHARPTR[1] EQL CAR_RET) OR (.CHARPTR[2] NEQ '''')
                                                                             SIGNAL (DBG$_INVCHRCON);
                                                                      TOKENBUFFER[0] = 3;
CH$MOVE(3, CHARPTR[0], TOKENBUFFER[1]);
```

D

```
DBGPARSER
V04-000
                                                                                                                       VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                                                                                                                                                        Page 201
(23)
                                                      CHARPTR = .CHARPTR + 3;
RETURN CREATE_OPERAND_TOKEN(TOKEN$K_IDENTIFIER, TOKENBUFFER);
  END:
                                             Do nothing for all other languages.
                                           [INRANGE, OUTRANGE]:
                                           TES:
                                        We have not found a valid token yet. This must therefore be a genuine
                                        syntax error, so we signal an appropriate error message.
                                     CHARPTR = .STARTPTR:
TOKENBUFFER[0] = 0:
INCR I FROM 0 TO 20 DO
                                           BEGIN
                                           IF .CHARPTR[.I] EQL CAR RET THEN EXITLOOP;
TOKENBUFFER[.I + 1] = .CHARPTR[.I];
                                           TOKENBUFFEREO] = .TOKENBUFFEREO] + 1;
                                      SIGNAL (DBG$_SYNERREXPR, 1, TOKENBUFFER);
                                      RETURN 0:
                                   END:
L1:5832
  INFO#250
 Referenced LOCAL symbol BACKUP_DIGIT_PTR is probably not initialized
                                                                                                               DBG$PLIT, NOWRT, SHR, PIC, 0
                                                                                 03117 P.AWZ:
03126
03134 P.AXA:
                                                      .ASCII
                                                                                                               <28>\DBGPARSER\<92>\LEXICAL_SCANNER 10\
                                                                                03126
03134 P.AXA:
0313A P.AXB:
03141 P.AXC:
03150
0315E P.AXD:
0316D
0317B P.AXE:
0318A
03199
0319C P.AXF:
                                                           54275457549
                                                                            4251C3C30C07
                                                                                                    .ASCII
                                                                                                               \%LABEL \
                                           2054154154
                                     53
453
453
55
55
                                                                                                    .ASCII
                                                                                                               <28>\DBGPARSER\<92>\LEXICAL_SCANNER 20\
                20
20
20
44
4F
49
                                                                                                               <28>\DBGPARSER\<92>\LEXICAL_SCANNER 30\
                                                                                                    .ASCII
                                                                                                    .ASCII \ DBGPARSER\<92>\DBG$LEXICAL_SCANNER 40\
                                                            00
                                                                                                    .ASCII \'\<0><0><0>
                                                                                                    .PSECT
                                                                                                               DBG$CODE, NOWRT, SHR, PIC, O
                                                                                                               DBG$LEXICAL SCANNER, Save R2,R3,R4,R5,R6,-R7,R8,R9,R10,R11
                                                                          OFFC 00000
                                                                                                    .ENTRY
```

D

DBGPARSER V04-000			F 13 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 202 (23)
	08	5E 00000000 FF 00000000 FF 9E 01	9E 00002 9A 00007 18: MOVZBL aCHARPTR, RO DF 0000E PUSHAL CHARTBL+2[RO] E1 00015 BBC #1, a(SP)+, 2\$ D6 00019 INCL CHARPTR 11 0001F BRB 1\$ D0 00021 28: MOVL CHARPTR, R4	5149
		9E 00000000 EF		5150
		56 57 64 57	9A 0002B MOVZBL (R4), R7 91 0002E CMPB R7, #13	5152
		00000000 FF	12 00031 BNEQ 38 7C 00033 CLRQ TERMINATOR_CODE	5160
	60	5B 00000000 EF 47	11 00039 BRB 8\$ DE 0003B 3\$: MOVAL CHARTBL[R7], R11	5160 5162 5171
	60	5B 00000000°EF47 6B 12 59 OC AC 58 01	DE 0003B 3\$: MOVAL CHARTBL[R7], R11 E1 00043 BBC #18, (R11), 10\$ DO 00047 MOVL TERM_LIST, R9 CE 0004B MNEGL #1, I 11 0004E BRB 9\$	5178
	55	50 00000000° EF	11 0004E BRB 9\$ 9E 00050 4\$: MOVAB TABLEBASE, RO C1 00057 ADDL3 (R9)[I], RO, TERMPTR	5180
		09 AS 08 AS	9E 00050 4\$: MOVAB TABLEBASE, RO C1 00057 ADDL3 (R9)[I], RO, TERMPTR 9A 0005C MOVZBL 8(TERMPTR), R10 29 00060 CMPC3 R10, 9(TERMPTR), (R4)	5188
		1A 01 AB 50 6A44 00000000°EF40	E9 00067 BLBC 1(R11), 5\$ 9A 0006B MOVZBL (R10)[R4], R0 DF 0006F PUSHAL CHARTBL[R0]	5193 5195
	35	00000000 EF 40	EO 00076 BBS #1, a(SP)+, 9\$ DF 0007A PUSHAL CHARTBLEROJ	5197
	2A	9E 02 02 02 10 A5 10 AC	DF 0007A PUSHAL CHARTBL[R0] E0 00081 BBS #2, a(SP)+, 9\$ E9 00085 5\$: BLBC 1(TERMPTR), 6\$ D5 00089 TSTL PAREN_NESTING	5201
	06	10 AC 21 65 09 57 01 A4	12 0008C BNEQ 9\$ E1 0008E 6\$: BBC #9, (TERMPTR), 7\$ 91 00092 CMPB 1(R4), R7 13 00096 BEQL 9\$	5205 5206
	000000	000' EF 02 A5 5A 5A	91 00092	5215 5216 5217
	90		F2 000AF 98: AORISS -4(R9) I 48	
****	4F OA	6B 000000006 00	F2 000AF 9\$: AOBLSS -4(R9), I, 4\$ E1 000B4 10\$: BBC #19, (R11), 14\$ BF 000B8 CASEB DBG\$GB LANGUAGE, #0, #10 000C0 11\$: .WORD 12\$-11\$,-	5178 5231 5234
0016 0016	0016 0016 0016	58 FC A9 6B 00 000000000 00 0016 0016 002D 0016 0016 0016	00000 12\$-11\$,- 12\$-11\$,- 12\$-11\$,- 13\$-11\$,- 12\$-11\$,-	
		00000000° EF	12\$-11\$,- 12\$-11\$,- 12\$-11\$ 9F 000D6 12\$: PUSHAB P.AWZ DD 000DC PUSHL #1	5260
	000000	00000000° EF 01 00028362 8F 0006 00 03	9F 000D6 12\$: PUSHAB P.AWZ DD 000DC PUSHL #1 DD 000DE PUSHL #164706 FB 000E4 CALLS #3, LIB\$SIGNAL	

					6 1 16-5 14-5	3 ep-1984 02 ep-1984 12	10:13 VAX-11 Bliss-32 V4.0-742 17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 203 (23)
		20	01 A4 07 02 00000000° EF	91 00	00EB	S: CMPE	14\$ R7, #45	5246
		3E	01 44	12 00 91 00 12 00 9E 00 04 00	00F0 00F2 00F6 00F8 00FF 0106	BNEC	14\$ 1(R4), #62	
	00000000	EF	02	CO 00	00F6 00F8	BNE G ADDL MOVA	14\$ #2, CHARPTR	5249 5250
		50		9E 00	00FF 0106	RET		
	00	AE 03	04 AC	DO 00 E8 00 9A 00 E9 00 9A 00	107 14 110C	S: MOVL	OPERAND_EXPECTED, 12(SP) 12(SP), 16\$	5273
		50	00000000 OOFB	9A 00	010c 0110 15 0113 16 011A 0122 17	S: BRW S: MOVZ	28\$ BL aCHARPTR, RO CHARTBL+2[RO], 15\$	1
		EE	00000000 EF40 00000000 EF	E9 00	11A 122 17	S: BLBC	LMARPIR	5276
		50	00000000 FF40	9A 00)128)12F	PUSH	I CHARTRI AZERNI	5276 5277
E8		9E	000000006 00	E0 00	12F 136 13A 141	BBS	#1, a(SP)+, 17\$ DBG\$GB_SET_BREAK_FLAG, 20\$ CHARPTR, RO	5285
	44	50 8F	000000000 00 000000000 EF 60	DF 00 E0 00 E9 00 91 00	141	MOVL	CHARPTR, RO	5285 5288
	4F	8F	10	12 00	1140	MOVI CMPE BNEG CMPE BNEG CMPE BE QL	19\$ 1(RO), #79	5289
		20	02 A0	12 00 91 00)153)155	BNEC	19\$ 2(RO), #32	5290
		28	02 A0	13 00	114E 1153 1155 1159 115B	BEQL	18\$ 2(R0), #40	
		51	01 A0 15 02 A0 06 02 A0 09 00000000 EF 7C	12 00 9E 00)15F)161 18	S: MOVA	19\$ CURLOC_TOKEN, R1	5292
	57	8F	7C		1168	S: CMPE	24\$ (RO), #87	5293
	48	8F	01 A0	12 00 91 00 12 00	16E	BNE	20\$ 1(RO), #72	5294
	45	8F	02 A0	12 00)16E)170)175)177	BNEG	20\$ 2(RO), #69	5295
	4E	8F	13	12 00	11/6	BNE	20\$ 3(RO), #78	5296
	7.	7	03 A0 0C 04 A0 74 04 A0 6E 00000000° FF	12 00	183	BNEG	20\$ 4(RO), #32	
		20	74	13 00)185)189)18B)18F)191 20	BEQL	26\$ 4(RO), #40	5297
		28	04 A0 6E	91 00	18F	BEQL	265	
		50	00000000'EF40	9A 00 E8 00 DF 00)191 20)198	S: MOVZ BLBS PUSH	L ACHARPTR, RO CHARTBLEROJ, 27\$	5331
50		9E 25	00000000°EF40	DF 00 E0 00 91 00	198 140 147	PUSH	CHARTBL[RO], 27\$ L CHARTBL+1[RO] #1, a(SP)+, 27\$ RO, #37	5332
		25	50 00	91 00	1 AB	CMPE	RO, #37 21\$	5333
		07	000000006 00 4E	12 00 91 00 12 00	1B0 1B7	CMPE	DBG\$GB_LANGUAGE, #7	5334
	5C	4A 8F	08 AC	E8 00	0180 0187 0189 0180 21	BBS CMPB BNEQ CMPB BNEQ BLBS S: CMPB	ADDRESS EXPRESSION, 27\$ (STARTPTR), #92	5337
		51	08 AC 66 09 00000000 EF	91 00 12 00 9E 00 11 00 91 00	10101	MOVA	22\$ CURVAL_TOKEN, R1	
	5E	8F	66 18	91 00	1CC 22	S: CMPB	(STARTPTR), #94	5338
		05	000000000 00	12 00 91 00	1103 1104 1100 1100 1102	ENE CMPB	25\$ DBG\$GB_LANGUAGE, #5	5339

DBGPARSER V04-000					H 13 16-Sep-1 14-Sep-1	1984 02:10 1984 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 20 (23
		0B 51 50	00000000° EF	12 0010 E9 0010 9E 0010 D0 001E	9 0B 0F 23\$: :6 24\$:	BNEQ BLBC MOVAB MOVL	23\$ ADDRESS_EXPRESSION, 25\$ PREVLOC_TOKEN, R1 R1, R0	534
		2E		91 001E	A 258:	CMPR	(STARTPTR), #46	534
		28	18 50	12 001E	F	BNEQ CMPB BEQL CMPB BEQL CMPB BEQL	27\$ RO, #46 27\$ RO, #92 27\$: 534
		2E	50	91 001F	4	CMPB	RO, #46	: 534
		5C 8F	00000000° EF	91 0016	9	CMPB	RO, #92	: 534
		50	00000000 EF	13 001F 9E 001F 04 0020	F 26\$:	MUVAB	CURLOC_TOKEN, RO	534
	000	000000° EF	00000000° EF 60 03	DO 0020 DO 0020 91 0021	7 275:	RET MOVL MOVL CMPB BEQL	STARTPTR, CHARPTR CHARPTR, RO (RO), #37	535 535
		F9 50 ED	0208	31 0021 E9 0021 9A 0022 E9 0022 D6 0022 D0 0023	A 29\$: D 30\$:	BLBC MOVZRI	STARTPTR, CHARPTR CHARPTR, RO (RO), #37 30\$ 68\$ 12(SP), 29\$ 1(RO), RO CHARTBL+1[RO], 29\$ CHARPTR	536 536
		59	00000000 EF	D6 0022	3	INCL	CHARPTR #1. TOKENLEN	536
		15 AÉ 50 0B	00000000 FF 00000000 EF40 00000000 EF40	90 0023 9A 0023 E8 0024 DF 0024	A 315:	BLBC INCL MOVL MOVB MOVZBL BLBS PUSHAL	#1, TOKENLEN #37, TOKENBUFFER+1 aCHARPTR, RO CHARTBL+1[RO], 32\$ CHARTBL+1[RO] #1, a(SP)+, 33\$ TOKENLEN RO, TOKENBUFFER[TOKENLEN] CHARPTR 31\$	536 536 537 537
	OF	9E	01	E1 0025	0	BBC	#1, a(SP)+, 33\$ TOKENLEN	
		14 AE49	00000000° EF	90 0025 90 0025 96 0025	6	MOVB	RO, TOKENBUFFER[TOKENLEN] CHARPTR	537
		14 AE 54	59 55	11 0026 90 0026 04 0026 CE 0026 11 0026	3 335:	MOVB	TOKENLEN, TOKENBUFFER INDEX #1, I 35\$	537 537 537 538 538
			23	9F 0026	E 348.	BRB MOVAB	35\$ TABLEBASE, RO	
59	5B 00	50 50 50 02 AB	00000000 EF 000000000 EF 01 AB 50 15 AE 05 6B 08 00000000 EF 55	C1 0027 9A 0027 2D 0028	5 E 12	MOVZBL CMPC5	TABLEBASE, RO PERCENT TABLE[I], RO, NAMEPTR 1(NAMEPTR), RO RO, 2(NAMEPTR), #0, TOKENLEN, TOKENBUFFER	538
		55	05 68	12 0028 9A 0028	A	BNEQ	35\$ (NAMEPTR), INDEX	539
	05		00000000' EF	11 0028 F2 0029	35\$:	BRB AOBLSS	36\$ PERCENT TABLE-4. 1. 34\$	539 539 538 540
019C 0239	05 08 002A 0231	00 002A 0229	55 0241 0221 0839	F2 0029 CF 0029 0029 0029 0029	71 35\$: 99 36\$: 90 37\$: ND	BRB AOBLSS CASEL .WORD	36\$ PERCENT TABLE-4, 1, 34\$ INDEX, #0, #8 67\$-37\$,- 38\$-37\$,-	540
							63\$-37\$,- 64\$-37\$,- 65\$-37\$,-	
							66\$-37\$,- 201\$-37\$	

						I 13 16-Sep- 14-Sep-	1984 02:10 1984 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 EDEBUG.SRCJDBGPARSER.B32;1	Page 205 (23)
				00000000 EF	9F 00	2AF	PUSHAB	P. AXC	; 5611
				00000000° EF 01 00028362 8F 03	DD 00	2B5 2B7	PUSHAB PUSHL PUSHL CALLS	#1 #164706	
		000000006	00	0217	FB 00	287 280 204	BRW	#3, LIB\$SIGNAL 67\$ R7	
			01	57	D4 00	207 38\$:	CLRI	R7	5424
			01	17	12 00	2CC	CMPL BNEQ INCL MOVC3	INDEX, #1	
15	AE	00000000	EF	06		200	MOVC3	R7 #6, P.AXA, TOKENBUFFER+1	5427
			59 52	06 06 8F 13 07 07	DO 00	209 200	MOVL MOVL BRB MOVC3	#6, P.AXA, TOKENBUFFER+1 #6, TOKENLEN #166386, ERRORMSG	5427 5428 5429 5424 5434 5435 5436
15	AE	00000000		13	DO 000	2E3	BRB	40\$	5424
1,	ME	0000000	EF 59	07	50 00	2E5 39\$: 2EE	MOVL	#7. TOKENLEN	5435
			50	000289EA 8F	28 00 00 00 00 00 9A 00	2F1 2F8 40\$:	MOVZBL	#166378, ERRORMSG acharptr. RO	: 5436
	09		9E	00000000°EF40	DF 00	2F F 306	PUSHAL BBS	CHARTBL+2[RO]	
	0,	*********		01 52	DD 00	30A	PUSHL	#7, P.AXB, TOKENBUFFER+1 #7, TOKENLEN #166378, ERRORMSG aCHARPTR, R0 CHARTBL+2[R0] #1, a(SP)+, 41\$ ERRORMSG #1, LIB\$SIGNAL aCHARPTR, R0 CHARTBL+2[R0] #1, a(SP)+, 42\$ CHARPTR	
		0000000G	00 50	00000000° FF	FB 00	30C 313 41\$:	CALLS MOVZBL	#1, LIB\$SIGNAL acharptr, RO	5444
	08		9E	00000000 EF40	DF 00 E1 00	31A	PUSHAL BBC	CHARTBL+2[RO]	
	-		"	00000000° FF	06 00	325	INCL	CHARPTR	5445
			50	00000000° FF 00000000° EF40	9A 00	325 328 320 42\$:	BRB MOVZBL	41\$ aCHARPTR, RO CHARTBL+1[RO] #1, a(SP)+, 43\$ ERRORMSG #1, LIB\$SIGNAL CHARPTR, RO (RO), #48	5450
	09		9E	00000000°EF40	DF 00	334 33B	PUSHAL BBS	CHARTBL+1[R0]	
		000000006		52	DD 00	33F	PUSHL	ERRORMSG	5452
		00000000	00 50 30	00000000 EF	E0 00 DD 00 FB 00 D0 00 91 00	348 438:	MOVL	CHARPTR, RO	5454
			30	000000000° EF	91 00: 12 00:	34F 352	CMPB BNEQ		
			50	00000000 EF 40	9A 00.	54	MOVZBL PUSHAL	1(RO), RO CHARTBL+1[RO] #1, @(SP)+, 44\$ CHARPTR	5455
	08		9E	01	DF 00 E1 00	55F	BBC	#1, a(SP)+, 44\$	
				00000000° EF	D6 00 11 00	565 569	INCL	A 4 4	5457
			50	00000000 FF 40	9A 00	6B 44\$:	BRB MOVZBL PUSHAL	aCHARPTR, RO	5463
	25	***********	9E 8F	01	DF 00	55F 563 569 56B 44\$:	BBC	aCHARPTR, RO CHARTBL+1[RO] #1, a(SP)+, 46\$ TOKENLEN, #255	
		000000FF	18	59 09	19 00	370 384 386 388 387 45\$:	BLSS	TOKENLEN, #255	5465
		000000006	00	52	DD 00	86	BLSS PUSHL CALLS	EKKURMSG	
				59	D6 00	8F 45\$:	INCL	#1, LIB\$SIGNAL TOKENLEN aCHARPTR, TOKENBUFFER[TOKENLEN]	5466 5467
		14 AE	149	00000000' FF	06 00 90 00 06 00 11 00	59A	INCL MOVB INCL	CHARPTR	5468
			03	00000000° EF C9 57	11 00	DAU	BRB BLBS	44\$ R7, 48\$	5468 5463 5476
				010F	31 00	SA2 46\$: SA5 47\$:	BRW	61\$: 2410
				00000000° FF	91 00	3A8 48\$: 3AF 3B1 3B8 3BA	CMPB BNEQ	acharptr, #46	
			50	00000000° EF	12 00 00 00 06 00 9A 00	581 588	MOVL	CHARPTR, RO	5483
			50	60	9A 00	BBA	MOVZBL	(RO), RO	

					13	13 -Sep-1 -Sep-1	984 02:10 984 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 206 (23)
0	9 000000FF	9E 8F	00000000°EF40	DF E1 D1	003BD 003C4 003C8		PUSHAL BBC CMPL	CHARTBL+1[RO] #1, a(SP)+, 49\$ TOKENLEN, #255	5484
	00000000G	00	09 52 01 59	19 DD FB D6 90 D6 D0 91	003C8 003CF 003D1 003D3 003DA	49\$: 50\$:	CMPL BLSS PUSHL CALLS INCL	ERRORMSG #1, LIB\$SIGNAL TOKENLEN	5486 5488 5489
	14 /	50 30	000000000 EF	90 06 00 91	003DC 003E1 003E7 003EE	51\$:	MOVB INCL MOVL CMPB	#46, TOKENBUFFER[TOKENLEN] CHARPTR CHARPTR, RO (RO), #48	5489 5490 5495
D	F	50 9E 50	00000000°EF40	12 9A DF EO 9A	003F1 003F3 003F7 003FE 00402		BNEQ MOVZBL PUSHAL BBS	52\$ 1(R0), R0 CHARTBL+1[R0] #1, a(SP)+, 51\$ aCHARPTR, R0	5496
9	1 000000FF	50 9E 8F	00000000° FF 00000000° EF40 01 59	9A DF E1 D1	00409	52\$:	BBS MOVZBL PUSHAL BBC CMPL	acharptr, ko Chartbl+1[RO] #1, a(SP)+, 47\$ TOKENLEN, #255	5503
	00000000G	00	09 52 01	19 DD FB	00414 0041B 0041D 0041F		PUSHL CALLS	53\$ ERRORMSG #1, LIB\$SIGNAL TOKENLEN	5505
	14 /	AE49	000000000 FF	90 96 11	00426 00428 00431 00437	53\$:	INCL MOVB INCL BRB	CHARPTR, TOKENBUFFER[TOKENLEN]	5506 5507 5508 5503 5533
0	8	50 9E	00000000 FF 40 00000000 EF 40 01	9A DF E1 D6	00439 00440 00447 0044B	54\$:	BRB MOVZBL PUSHAL BBC INCL	52\$ aCHARPTR, RO CHARTBL+2[RO] #1, a(SP)+, 55\$ CHARPTR	5533
		56 50	00000000 EF	11 00 9A	00451 00453 0045A	55\$:	BRB MOVL MOVZBL	CHARPTR, STARTPTR	5536 5542
0		9E	00000000°EF40 02 10 AE 18 AE 02	DF E1 9F 9F	00461 00468 0046C 0046F 00472 00477		PUSHAL BBC PUSHAB PUSHAB	CHARTBL+1[RO] #2, a(SP)+, 56\$ TOKEN TYPE TOKENBUFFER	5544
	0000V	50 50	00000000 FF 00000000 EF40	FB 11 9A DE	00472 00477 00479 00480	56\$:	PUSHAB CALLS BRB MOVZBL MOVAL	MZ, SCAN_GOOLED_SIKING	5553
0	4	50 08 60 60	00000000° EF	9A DE E8 E0 E1 D6	00480 00488 0048B 0048F	57\$:	BBS BBC INCL	aCHARPTR, RO CHARTBL[RO], RO (RO), 57\$ #1, (RO), 57\$ #2, (RO), 58\$ CHARPTR	5554 5555 5557
5	9 00000000	EF	DE 56 0D 000289BA 8F	C3	00499 0049B 004A3	58\$:	SURI 3	STARTETE CHARPER TOKENIEN	5559 5560
15 A	00000000G	00 66 AE	01 59 59	DD FB 28 90 31	0048F 00499 0049B 004A5 004A5 004B2 004BB 004BE 004C5	59\$: 60\$: 61\$: 62\$:	CALLS MOVC3 MOVB BRW	60\$ #166330 #1, LIB\$SIGNAL TOKENLEN, (STARTPTR), TOKENBUFFER+1 TOKENLEN, TOKENBUFFER 201\$	5561 5562 5577 5584
			00000000° 0918	9E	004BE 004C5	63\$:	MOVAB RET	201\$ RADIX_OP_DEC, RO	
		50	00000000° EF	9E 04	004C6 004CD	64\$:	MOVAB RET	RADIX_OP_HEX, RO	5591

DBGPARSER V04-000			K 13 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 207 (23)
		50 00000000° EF		; 5598
		50 00000000° EF	9E 004D6 66\$: MOVAB RADIX_OP_BIN, RO	5605
	00000000	50 00000000° FF 00000000° EF40 9E 02	04 004DD D0 004DE 67\$: MOVL STARTPTR, CHARPTR 9A 004E5 68\$: MOVZBL aCHARPTR, RO DF 004EC PUSHAL CHARTBL+1[RO] E1 004F3 BBC #2, a(SP)+, 69\$ 9F 004F7 PUSHAB TOKEN TYPE 9F 004FA PUSHAB TOKENBUFFER FB 004FD CALLS #2, SCAN QUOTED_STRING 9F 00505 PUSHAB TOKENBUFFER DD 00505	5621 5634
	14	9E 02 10 AE 18 AE	DF 004EC PUSHAL CHARTBL+1[R0] E1 004F3 BBC #2, a(SP)+, 69\$ 9F 004F7 PUSHAB TOKEN TYPE 9F 004FA PUSHAB TOKENBUFFER FB 004FD CALLS #2, SCAN_QUOTED_STRING	5637
	0000v	10 AE 18 AE 02 14 AE 14 AE 08D0 50 000000000° FF	FB 004FD CALLS #2, SCAN_QUOTED_STRING PF 00502 PUSHAB TOKENBUFFER DD 00505 PUSHL TOKEN_TYPE 31 00508 BRW 202\$ 9A 0050B 69\$: MOVZBL @CHARPTR, R0	5638
	03	9E 00000000°EF40	DF 00512	5655
		05 08 AC 50	31 0051D 70\$: BRW 107\$ E9 00520 71\$: BLBC ADDRESS_EXPRESSION, 72\$ 91 00524 CMPB RO, #46	5656
		09 000000006 00 00 00	NA NOSOO 778. CIDI STATE TAINEY	5669 5670
		OA 00000000° EF	D1 00534 CMPL EXPRESSION_RADIX, #10 13 0053B BEQL 73\$	5671
		58 1F	DO 00543 MOVL EXPRESSION_RADIX, RO D1 0054A CMPL RO, #16	5673 5675 5676
		57 50 00000000° EF 10 03 57 08 50 03	12 0054D BNEQ 74\$ D0 0054F MOVL #5, NUMBER_KIND D1 00552 74\$: CMPL R0, #8 12 00555 BNEQ 75\$	5678 5680
		57 02 50 03	DO 00557 MOVL #11, NUMBER_KIND D1 0055A 75\$: CMPL RO, #2 12 0055D BNEQ 76\$	5682 5684
		57 02 50 57 5A 52 000000000 EF 50 000000000 EF	DO 00557 D1 0055A 75\$: CMPL RO, #2 BNEQ 76\$ D0 0055F MOVL #10, NUMBER_KIND D0 00562 76\$: MOVL NUMBER_KIND, BACKUP_NUMBER_KIND D0 00565 77\$: MOVL CHARPTR, R2 9A 0056C MOVAL CHARTBL[RO], R3 EF 00577 P1 00570 P1 00586 P1 00586 P1 00586 P1 00586 P1 00587 P1 00588 P1 0	5686 5688 5703
04 AE	63	53 00000000 EF 40 04	DE 0056F MOVAL CHARTBL[RO], R3 EF 00577 EXTZV #4, #4, (R3), CLASS	
		09 00000000G 00 0E	91 0057D CMPB DBG\$GB_LANGUAGE, #9 12 00584 BNEQ 78\$	5704
		2E 50 09 2E 01 A2	12 00589 BNEQ 78\$ 91 00588 CMPR 1(R2) #46	5706
		03 04 00000000°FF48	12 00584 91 00586 12 00589 91 0058B 91 0058B 12 0058F D4 00591 DF 00594 78\$: PUSHAL @STATE_TABLE[STATE_INDEX] 9A 0059B 13 0059E D1 005A0 13 005A0 13 005A6 11 005A8 BRB 78\$	5708 5717
	04 '	50 9E 0A AE 50	13 0059E BEQL 79\$	5710
	04	04	13 005A4 BEQL 79\$	5719
		ÉÀ	06 005A6 INCL STATE_INDEX 11 005A8 BRB 78\$: ""

*

DBGPARSER V04-000			L 13 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 208
08 AE 58 0045 008F 00D2 00B3	9E 9E 111 0040 0059 0024 0072	08 10 01 01 03B 00000000°FF48 10 01 01 08 01 004A 000D 005E 005E 0054 0127	DF 005AA 79\$: EF 005B1 DF 005B7 EF 005B2 CF 005C3 005C8 80\$: 005D0 005D8 005E8 DF 005B8 DF 005C8 DF 005C8	5723 5724 5730
	000000000	6E 52	97\$-80\$,- 106\$-80\$,- 87\$-80\$ 9F 005EC 81\$: PUSHAB P.AXD DD 005F2 PUSHL #1 DD 005F4 PUSHL #164706 FB 005FA CALLS #3, LIB\$SIGNAL 11 00601 BRB 93\$ D0 00603 82\$: MOVL R2, BACKUP_DIGIT_PTR 11 00606 BRB 93\$ D0 00608 83\$: MOVL #6, NUMBER_KIND 11 00608 BRB 93\$ D0 00600 84\$: MOVL R2, BACKUP_DIGIT_PTR	5944 5745
			DO 00608 83\$: MOVL #6, NUMBER_KIND 11 0060B BRB 93\$ DO 0060D 84\$: MOVL R2, BACKUP_DIGIT_PTR 11 00610 BRB 92\$ DO 00612 85\$: MOVL #7, NUMBER_KIND 11 00615 BRB 93\$	5752 5761 5762 5769
		57 07 57 08 57 08 57 07 57 09 57 09 6E 52 05 57 08	DO 00617 86\$: MOVL #8, NUMBER_KIND	5775 5781 5787
		6E 52 57 68	11 0061A D0 0061C 87\$: MOVL #15, NUMBER_KIND 11 0061F BRB	5796 5797
		OB 57 63 0A 57 5E 12 6E 52 05 57	D1 0062E CMPL NUMBER_KIND, #11 13 00631 BEQL 99\$ D1 00633 CMPL NUMBER_KIND, #10 13 00636 BEQL 99\$ 11 00638 BRB 91\$	5798 5799
			DO 0063A 90\$: MOVL R2, BACKUP_DIGIT_PTR D1 0063D CMPL NUMBER_KIND, #5 13 00640 BEQL 92\$ D1 00642 CMPL NUMBER_KIND, #11	5801 5811 5812
		OB 57 OA 57 O5	01 00642 CMPL NUMBER_KIND, #11 13 00645 BEQL 92\$ D1 00647 CMPL NUMBER_KIND, #10 13 0064A BEQL 92\$	5814

		16	13 -Sep-1984 02:10:13 -Sep-1984 12:17:30	VAX-11 Bliss-32 V4.0-742 CDEBUG.SRCJDBGPARSER.B32;1	Page 209 (23)
	5A				; 5816
	5A	03 11 0064F 06 00 00651	92\$: BRB 93	4. BACKUP_NUMBER_KIND S. BACKUP_NUMBER_KIND	
	57 6E 00000FF 8F	01 C1 0065A 56 C3 00662 59 D1 0066A 5E 14 00671	94\$: MOVL BA ADDL3 #1 SUBL3 ST 95\$: CMPL TO	ACKUP NUMBER KIND, NUMBER KIND I, BACKUP DIGIT PTR, CHARPTR TARTPTR, CHARPTR, TOKENLEN OKENLEN, #255	5818 5730 5831 5832 5833 5834
59	52	69 11 00673 56 C3 00675	BRB 10	15\$ TARTPTR, R2, TOKENLEN	5835
	05	FF 11 00679 57 D1 0067B	BRB 95	SUMBER_KIND, #5	5835 5847 5848 5861
	0B	F5 13 0067E 57 D1 00680	BEQL 96	JMBER_KIND, #11	5862
	0A	F0 13 00683 57 D1 00685	BEQL 96 CMPL NU	SSUMBER_KIND, #10	5863
	57	FF 11 00679 57 D1 00678 F5 13 0067E 57 D1 00680 F0 13 00683 57 D1 00685 EB 13 00688 0E D0 0068A E6 11 0068D 000° EF D1 0068F 57 13 00696	BEQL 96	\$ 14. NUMBER_KIND	:
	10 00000	TOUGHT ET UT OUGGE	98\$: CMPL EX	PRESSION_RADIX, #16	5865 5867 5889
		57 13 00696 0B 11 00698	995: BEQL 10 BRB 10	06 \$ 01 \$:
04 09	08 63 63	63 E8 0069A 01 E0 0069D	100\$: BLBS (R	(3) . 101\$	5892 5909 5910
	000000° EF	02 E1 006A1 56 D0 006A5	101\$: BBC #2	(R3), 101\$ 2, (R3), 102\$ [ARTPTR, CHARPTR	5911
	05	4A 11 006AC 57 D1 006AE 0D 13 006B1	102\$: CMPL NU	JMBER_KIND, #5	5911 5914 5913 5918
	0B	0D 13 006B1 57 D1 006B3 08 13 006B6	CMPL NU	JMBER_KIND, #11	5919
	0A	08 13 00686 57 01 00688	CMPL NU	JMBER_KIND, #10	5920
50.00	57		BEQL 10 MOVL #1 103\$: SUBL3 ST	3\$ 4, NUMBER_KIND	5922
59 000	000000° EF 0000FF 8F	0E DO 006BD 56 C3 006C0 59 D1 006C8 0D 15 006CF 9C2 8F DD 006D1 01 FB 006D7	103\$: SUBL3 ST	A, NUMBER KIND (ARTPTR, CHARPTR, TOKENLEN (KENLEN, #255) (A) (A) (A) (A) (A) (A) (A) (A) (A) (A	5922 5924 5925
000	00028	9C2 8F DD 006D1	104\$: BLEQ 10	66338	
	000000G 00 14 AE 66	59 90 006DE	105\$: CALLS #1 105\$: MOVB TO MOVC3 TO PUSHAB TO	KENLEN, TOKENBUFFER	5926
15 AE	00	59 D1 006C8 0D 15 006CF 0D 15 006CF 01 FB 006D7 59 90 006DE 59 28 006E2 14 AE 9F 006E7 57 DD 006EA 06EC 31 006EC	PUSHAB TO	KENBUFFER	5926 5927 5928
	00000	06EC 31 006EC	PUSHL NU BRW 20	JARPTR	5051
	00000	000° EF D6 006EF	BRW 77		5951 5695 5963
	57 00000 50 03 00000	67 9A 006FF	107\$: MOVL CH	ARPTR, R7 R7), R0 IARTBL[R0], 108\$	3903
	03 00000	0000 EF 40 E8 00702 0147 31 0070A 57 D7 0070D 0000 EF D0 0070F	BLBS CH BRW 12	26\$	507/
	51 00000 50	0000' EF DO 0070F	108\$: DECL EN MOVL CH 109\$: MOVZBL (R	IDPTR	5974 5977
0E	00000	000'EF40 DF 00719	PUSHAL CH	II), RÓ IARÍBL[RO] 2. a(SP)+, 110\$ 2. ENDPTR	
ÜE.	9E 57	02 E1 00720 51 D0 00724	BBC #2 MOVL R1	ENDPTR	: 5980

. 59

						16 14	13 -Sep-19 -Sep-19	84 02:10 84 12:17	:13 VAX-11 Bliss-32 V4.0-742 P.:30 [DEBUG.SRC]DBGPARSER.B32;1	age 210 (23)
	26		9E	00000000'EF40	DF E1	00727 0072E		PUSHAL BBC	CHARTBLEROJ #1. a(SP)+. 1115	: 5981
	-		51	00000000 EF	D6 D0 9A		110\$:	INCL MOVL MOVZBL	W1, a(SP)+, 111\$ CHARPTR CHARPTR, R1 (R1), RO	5984 5985
	C9		9E	00000000°EF40	DF	00742		PUSHAL BBS	CHARTBL[RO] #1, a(SP)+, 109\$	
	BE		9E	00000000°EF40	DF EO	0074D 00754	1	PUSHAL BBS	CHARTBL[RO] #2. a(SP)+. 109\$	5986
	59	00000000° 0000000FF	EF EF 8F	01 Å7 56 59	9E C3 D1	00760 00768	111\$:	MOVAB SUBL3 CMPL	1(R7), CHARPTR STARTPTR, CHARPTR, TOKENLEN TOKENLEN, #255	5990 5995 5996
		0000000G	00	00028982 8F	DD FB	0076F 00771 00777		PUSHL CALLS	112\$ #166274 #1, LIB\$SIGNAL	
15	AE	14	66 AE	59	28		112\$:	MOVE3 MOVB	TOKENLEN, (STARTPTR), TOKENBUFFER+1 TOKENLEN, TOKENBUFFER	5997 5998
		000000006	3B 00	00000000° EF 56 09	E9	00787 0078E		BLBC	STARTPTR, DBG\$GL_UPCASE_COMMAND_PTR	; 6010 ; 6013
		0000000G	00	09 56 15	19 01	00795		BLSS	STARTPTR, DBG\$GL_UPCASE_COMMAND_PTR+4	: 6014
				00000000° EF	15 9F DD	0079E 007A0 007A6	113\$:	PUSHAB PUSHL	114\$ P.AXE #1	6016
		000000006	00	00028362 8F	DD	007A8 007AE		PUSHL	#164706 #3, LIB\$SIGNAL	
	50		56	00028362 8F 03 00000000G 00 0000000G 00 59	C3	007B5 007BD	114\$:	SUBL3 ADDL2	DBG\$GL_UPCASE_COMMAND_PTR, STARTPTR, RO DBG\$GL_ORIG_COMMAND_PTR, NEW STARTPTR TOKENLEN, (NEW STARTPTR), TOKENBUFFER+1	6018
15	AE		60	5A	28	007C4 007C9	115\$:	MOVC3 CLRL	BEST_TOKEN_FOUND	: 6020
			55 54	00000000° EF	CE	007CB 007D2		MOVL	IDENT_OPERATOR_TABLE, R5	6029
	58		50	00000000° EF	11 9E	007D5 007D7	116\$:	MOVAB ADDI 3	121\$ TABLEBASE, RO (RS)(I) RO TOKEN	6031
	00	OD	50 50 A8	OC A8	9A 2D	007DE 007E3 007E7		ADDL3 MOVZBL CMPC5	TABLEBASE, RO (R5)[I], RO, TOKEN 12(TOKEN), RO RO, 13(TOKEN), #O, TOKENLEN, TOKENBUFFER+1	6032
				15 AE		007E7 007ED 007EF		BNEQ	1215	
			0C	0C AE 68 03	12 E9 91	007F1 007F5		BLBC	12(SP), 119\$ (TOKEN), #2	6036
			05	0549 0C AE 68	12 31 E8	007F8 007FA 007FD	117\$: 118\$:	BNEQ BRW BLBS	118\$ 193\$ 12(SP), 120\$	6038
			05	F4	91	00804	1185:	BLBS CMPB BNEQ	(TOKEN) . #2	:
	c9		5A 54	FC A5 5A 36 5A	DO F2 D5 12	00806 00809 0080E 00810	120\$: 121\$:	MOVL AOBLSS TSTL	117\$ TOKEN, BEST_TOKEN_FOUND -4(R5), I, T16\$ BEST_TOKEN_FOUND 124\$	6044 6029 6050
			55	00000000 EF	DO CE	00812 00814 0081B		BNEQ CLRL MOVL MNEGL	BEST TOKEN_FOUND	6057 6058
				00000000° EF	11 9E	0081E 00820	122\$:	BRB	#1. I 123\$ TABLEBASE, RO (R5)[I], RO, TOKEN 8(TOKEN), RO	6060
	58		50 50 50	08 6544	9A	00827 0082C		ADDL3 MOVZBL	8(TOKEN), RO	6061

DBGPARSER V04-000									12	14 5-Sep-1 5-Sep-1	984 02:10 984 12:17	13 VAX-11 Bliss-3 30 [DEBUG.SRC]DBC	32 V4.0-742 SPARSER.B32;1	Page 21 (23
	59		00	09	A8	15	50 AE 07	20	00836		CMPC5		OKENLEN, TOKENBUFFER+1	: 606
					BC	ОС	O7	12 E8 D0 F2	00838 0083A 0083E		BNEQ BLBS	123\$ 12(SP), 117\$		606
			DA		BC 5A 54	FC	AE 58 54 53	F2	0083E 00841	1238:	MOVL	TOKEN, BEST_TOKEN_FO -4(R5), I, T22\$ BEST_TOKEN_FOUND 125\$ 154\$	DUND	606 606 607
							03	13	00846 00848	1245:	BEQL	BEST_TOKEN_FOUND		: 607
							01BF	05	0084A 0084D	1258:	BRW TSTL	TUKENLEN		: 607
							01BF 59 03 0582	31	0084F 00851 00854		BEQL	126\$ 201\$		1
			^3		50	000000000	EF40	9A DF	00854 0085B	126\$:	PUSHAL	CHARTBL+1[RC]		: 608
			03		9E		00F1	E0	0085B 00862 00866 00869		BBS BRW	CHARPTR, RO CHARTBL+1[RO] #3, a(SP)+, 127\$ 143\$		1
			00			00000000	EF 40	9A DF	00869 00870 00877	127\$:	PUSHAL	aCHARPTR, RO CHARTBL+1[RO] #4, a(SP)+, 128\$: 610
			08		9E	00000000	O4 EF	E1	0087B		BBC	CHARPIR		: 610
					50		FF.	11 9A	00881	128\$:	BRB MOVZBL	127\$ aCHARPTR, RO		: 610
			08		9E	00000000.	05	DF E1 D6	00891		PUSHAL BBC	CHARPTR, RO CHARTBL+1[RO] #5, a(SP)+, 129\$ CHARPTR		410
					50	00000000	EF E6 FF	11 9A	0088A 00891 00895 0089B 0089D	1298:	INCL BRB	128\$ acharptr, ro chartbl+1[ro]		610
			08		9E	00000000	EF40	DF E1	008A4 008AB	1278:	MOVZBL PUSHAL BBC	CHARTBL+1[RO]		. 610
			00		~	00000000	EF E6	D6	008AF 008B5		INCL BRB	#6, a(SP)+, 130\$ CHARPTR 129\$: 610
					56	00000000	EF 06	D1 12	008B7	130\$:	CMPL BNEQ	CHARPTR, STARTPTR		610
			59	00000000	FF	00000000.	EF 56	D6	00800	1315:	INCL SUBL3	CHARPTR STARTPTR CHARPTR 1	OKENI EN	611
				0000000° 000000FF	EF 8F		59 0D 8F	C3 D1 15	008CE 008D5		CMPL	CHARPTR STARTPTR, CHARPTR, 1 TOKENLEN, #255 132\$ #166306	OKENEEN	611
				0000000G	00	000289A2	8F 01	15 DD FB 90	008D7 008DD 008E4		PUSHL	#166306 #1. LIB\$SIGNAL		•
		15	AE	14	00 AE 66		59	90	008E4 008E8	132\$:	BLEQ PUSHL CALLS MOVB MOVC3	TOKENLEN, TOKENBUFFE TOKENLEN, (STARTPIR)	R TOKENBUFFER+1	611
						00000000	5A EF	28 04 00 CE	008E8 008ED 008EF 008F6 008F9		CLRL	V1, LIB\$SIGNAL TOKENLEN, TOKENBUFFE TOKENLEN, (STARTPTR) BEST TOKEN FOUND OPCHĀR_OPERATOR_TABL	E. R5	612
					55		01 3A	CE 11	008F6 008F9		MNEGL	136e		
			58		50	00000000.		9E	008FB 00902	133\$:	MOVAB ADDL3	TABLEBASE, RO (R5)[I], RO, TOKEN		612
					26 50 A8	01 08 00	A8 AC	9E C1 E8 E8 9A	00902 00907 0090B 0090F		BLBS BLBS	1 (TOKEN), 1348 ADDRESS_EXPRESSION,	139\$	613
	59		00	OD	50 A8	00	6544 A8 A8 50 AE 18 A683	9A 2D	0090F 00913	1348:	MOVZBL CMPC5	12(TOKEN), RO RO, 13(TOKEN), #0, 1	139\$ OKENLEN, TOKENBUFFER+1	613
						15	AE 18	12	00919			139\$ 12(SP), 137\$		•
					02 02	00	AE 68	12 E9 91	00921		BNEQ BLBC CMPB	(TOKEN). #2		613
							0410	12	00924	135\$:	BNEQ BRW	136\$ 193\$		1

AOBLSS

C3

DBGPARSER V04-000							1	0 14 6-Sep-1 4-Sep-1	984 02:10 984 12:17	:13	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page (213
				50		04	00A08	1548:	TSTL BEQL MOVL RET	BEST 155\$ BEST	_TOKEN_FOUND, RO	. 6	5223
03C2 00F9	000	0A 3C2 3C2 3C2	00000000	00 00 00 03 03 02 02 02 07	00000000G 00 03C2 03C2 01BC	8F	00A10 00A17 00A1F 00A27 00A2F	155\$: 156\$: 157\$:	MÖVL CASEB .WORD	158\$- 203\$- 203\$- 203\$- 203\$- 167\$-	TPTR, CHARPTR GB_LANGUAGE, #0, #10 -157\$,157\$,157\$,157\$,157\$,157\$,157\$,157\$,157\$,-	6	5224
				1B	03A9 08 AC 0C AE	E9	00A35 00A38	158\$:	BRW BLBC	203\$ 203\$ ADDRI	-157\$,- -157\$,- -157\$ ESS_EXPRESSION, 159\$ P),-159\$ RPTR, #46	: 6	5248
				SE J	OC NE	E9 91	00A3C 00A40		BRW BLBC BLBC CMPB BNEQ INCL	12(SI 0CHAI	P), 159\$ RPTR, #46	6	5249
				50	00000000° EF		OGA4F		MUVAB	FORT	PTR RAN_INDIRECT_TOKEN, RO	: 6	525
				50	00000000 EF	04	00A56 00A57 00A5E	159\$:	RET MOVL INCL MOVZBL		PTR, RO	:	526
				50 08	00000000 EF 000000000 EF 000000000 EF 60	9A E9	00A60 00A63		MOVŽBL BLBC INCL	(RO)	, RO TBL+1[RO], 160\$ PTR		174
			00000000	EF	E4 02	11	00A71 00A73	160\$:	BRB	1595			526°
		55	000000000	EF 54	00000000 EF 64 02 56 01 20 00000000 EF	C3	00A63 00A6B 00A71 00A73 00A7A 00A82 00A87		ADDL2 SUBL3 MNEGL	STAR #1 162\$	CHARPTR TPTR, CHARPTR, R5 I	. 6	26
		58		50 50	00000000 EF 44	9E	00A87 00A8F	161\$:	MOVAB ADDL 3	TABLE	BASE, RO	6	266
55		00	OD	50 A8	OC A8	9A 2D	00A8E 00A97 00A9B 00AA1		ADDL3 MOVZBL CMPC5	12(T)	EBASE, RO RAN SPECIAL_OPTBL[1], RO, TOKEN DKEN), RO 13(TOKEN), #0, R5, (STARTPTR)	6	267
		D8		54	00000000 66 03 029F	12 31 F2	00AA2	162\$:	BNEQ BRW AOBLSS	162\$ 193\$ FORTE	RAN_SPECIAL_OPTBL-4, I, 161\$		264
			08	50 AE 5A	000000000 EF 000000000 EF FC AO	DO	00ABF 00ABF 00ABB		MOVL MOVL MNEGL	PRID	TBL, RO 0), 8(SP)	6	264
				50	00000000 FF4A	11 9E	00ABE 00ACO	163\$:	BRB	1645 TABLE	BASE, RO	6	279
		57	00000000	50	00000000°FF4A	94	00AC7 00AD0		MOVZBL	8(PR	EBASE, RO DTBL[1], RO, PRID ID), R1 IPTR, CHARPTR, RO P(PRID), #O, RO, (STARTPTR)		280 281 280
50		00	00000000	A7	51	20	00ADC 00AE2		SUBL3 CMPC5			6	280
	15	AE	00	50 A7 AE	08 A7	12 9A	00AES		MOVZBL	164\$ 8(PR)	ID), RO	6	284
	.,	ML.	09 14	AE	08 A7	90	00AE9 00AEF		MOVE3 MOVB	8(PR	P(PRID), TOKENBUFFER+1 ID), TOKENBUFFER	: 6	285

	E 14 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 214 (23)
C4 5A 08 AE 02DE 02DE 02DE 66 F8 000000000 EF 50 000000000 EF	31 00AF4 F2 00AF7 164\$: AOBLSS 8(SP), I, 163\$ E9 00AFC BLBC 12(SP), 166\$ 31 00B00 165\$: BRW 203\$ 91 00B03 166\$: CMPB (STARTPTR), #46 12 00B06 BNEQ 165\$ 9E 00B08 MOVAB 1(R6), CHARPTR 9E 00B10 MOVAB FORTRAN_DOT_TOKEN, RO 04 00B17 RET	6286 6277 6294 6297 6298
50 00000000° EF 26 00000000° EF 31 00000000° EF 26 00000000° EF 0E 50 00000000° EF	91 00B18 167\$: CMPB	6316 6319 6320
50 00000000° EF 08 0C AE 50 00000000° EF	9E 00B36 MOVAB C_AND_TOKEN, RO 04 00B3D RET E9 00B3E 168\$: BLBC 12(SP), 169\$ 9E 00B42 MOVAB C_ADDR_OF_TOKEN, RO 04 00R49 RET	6323 6324 6327 6328
2B 00000000° FF 2A 00000000° EF 2B 00000000° FF 13 00000000° EF 00028F78 8F	91 00B52 170\$: CMPB	6335 6338 6339 6342 6357
50 00000000° EF 2D 00000000° FF 75 00000000° EF	DD 00B70 PUSHL #167800 FB 00B76 CALLS #1, LIB\$SIGNAL 9E 00B7D 171\$: MOVAB C_ADD_TOKEN, R0 04 00B84 RET 91 00B85 172\$: CMPB aCHARPTR, #45 12 00B8C BNEQ 179\$ D6 00B8E INCL CHARPTR 91 00B94 CMPB aCHARPTR, #45	6357 6361 6368 6371 6372
2D 00000000° FF 000000000° EF 00028F78 8F 01 3E 00000000° FF 0E	91 00B94 CMPB aCHARPTR, #45 12 00B9B BNEQ 173\$ D6 00B9D INCL CHARPTR DD 00BA3 PUSHL #167800 FB 00BA9 CALLS #1, LIB\$SIGNAL 91 00BB0 173\$: CMPB aCHARPTR, #62 12 00BB7 BNEQ 175\$ D6 00BB9 INCL CHARPTR 9E 00BBF 174\$: MOVAB C_ARROW_TOKEN, R0	6372 6375 6389 6393
3E 00000000 FF 000000000 EF 50 00000000 EF 50 00000000 EF	06 00BB9	6396 6397 6400 6402 6404
2A 00000000° FF 03 0C AE 00000000° EF 51 00000000° EF 50 61	04 00BDA RET CMPB aCHARPTR, #42 12 00BE2 BNEQ 179\$ E8 00BE4 BLBS 12(SP), 178\$ 31 00BE8 BRW 185\$ D6 00BEB 178\$: INCL CHARPTR D0 00BF1 MOVL CHARPTR, R1 9A 00BF8 MOVZBL (R1), R0	6414 6420 6423 6424

0E 26	03 00000000°EF40 0108 57 FE A1 51 00000000° EF 50 00000000°EF40 9E 02 57 51	E8 00BfB 31 00C03 179\$: BRW 203\$ 9E 00C06 180\$: MOVAB -2(R1), ENDPTR D0 00C0A MOVL CHARPTR, R1 9A 00C11 181\$: MOVZBL (R1), R0 DF 00C14 PUSHAL CHARTBL[R0] E1 00C1B BBC #2, a(SP)+, 182\$ D0 00C1F MOVL R1, ENDPTR DF 00C22 PUSHAL CHARTBL[R0] PSHAL CHARTBL[R0] PSHAL CHARTBL[R0]	6435 6438
C9 BE 000000000° 59 00000000°	9E 00000000 EF 40 01 51 00000000 EF 61 00000000 EF 40 01 02 02 03 03 00000000 EF 843 000028982 F8 833	D6 00C2D 182\$: INCL CHARPTR D0 00C33 MOVL CHARPTR, R1 9A 00C3A MOVZBL (R1), R0 DF 00C3D PUSHAL CHARTBL[R0] E0 00C44 BBS #1, a(SP)+, 181\$ PUSHAL CHARTBL[R0] BBS #2, a(SP)+, 181\$ 9E 00C53 183\$: MOVAB 1(R7), CHARPTR C3 00C5B SUBL3 STARTPTR, CHARPTR, TOKENLEN D1 00C63 CMPL TOKENLEN, #255 14 00C6A BGTR 184\$ 31 00C6C BRW 60\$ DD 00C6F 184\$: PUSHL #166274 31 00C75	6441 6442 6445 6446 6447 6451 6456 6457
		D6 00C78 185\$: INCL CHARPTR 9E 00C7E MOVAB RPG_MULTIPLY_TOKEN, RO 04 00C85 RET 91 00C86 186\$: CMPR ACHARPTR #39	6468
000000006 68 00000000° 58	03 0C AE 0108 00 01 58 50 6F 0E 57 000000000 EF 50 000000000 EF 51 6B 54 000000000 EF 51 6B	13 00C8D 31 00C8F E9 00C92 187\$: BLBC 12(SP), 188\$ 31 00C96 BRW 198\$ DD 00C99 188\$: PUSHL #8 CALLS #1, DBG\$GET_TEMPMEM MOVL RO, TOKEN PO 00CAD MOVL #1, INDEX 9E 00CBO 189\$: MOVAB TABLEBASE, RO C1 00CB7 ADDL3 ADA TICK_TOKEN, (TOKEN) 9A 00CC0 MOVZBL (NAMEPTR), R1 9A 00CC3 MOVZBL (NAMEPTR), R0 DO 00CC6 MOVL CHARPTR, R4 CMPC5 R1, 1(NAMEPTR), #0, R0, 1(R4)	6485 6494 6500 6501 6505 6508 6510
50 00000000. 00 00 00000000.	A8 50 A8 50 AE 6B 5A 50 59 0D A8 EF 01	12 00CD5 BNEQ 194\$ 9A 00CD7 MOVZBL (NAMEPTR), R0 9E 00CDA MOVAB 1(R4)[R0], CHARPTR BO 00CE3 MOVW INDEX, 6(TOKEN) 9A 00CE7 MOVZBL (NAMEPTR), R0 D6 00CEA INCL R0 90 00CEC MOVB R0, 12(TOKEN) 9A 00CEO MOVZBL (NAMEPTR) 12(SP)	6514 6515 6516 6517
	BE 00000000° 0000000° 0000000° 0000000° 000000	00000000	C9

DBGPARSER V04-000					6 14 16-Sep- 14-Sep-	1984 02:10 1984 12:17	:13 VAX-11 BLiss-32 V4.0-742 :30 CDEBUG.SRCJDBGPARSER.B32:1	Page 216 (23)
5A	20	01 AE	59 5A 0C AE	06	00D07 00D09 00D0B	INCL DECL MOVCS	R9 R10 12(SP), 1(NAMEPTR), #32, R10, (R9)	1
		50	69		00013	MOVZBL		6527
	08	98	01	9A DF E1 D6	00013 190\$: 0001A 00021 00025 0002B 0002D 191\$: 00034 00036 00040	PUSHAL BBC INCL	aCHARPTR, RO CHARTBL+2[RO] #1, a(SP)+, 191\$ CHARPTR 190\$	6528
		28	00000000° EF	11	0002B 0002D 191\$:	BRB	190S aCHARPTR #40	6530
		01 A8	000	12	00034 00036	BRB CMPB BNEQ BISB2	aCHARPTR, #40 192\$ #8, 1(TOKEN) CHARPTR 193\$	
		01 45	00000000° EF 04 08 58	88 06 11	00D3A 00D40	BRB	CHARPTR 193\$	6533 6534 6537 6537
		01 A8	58	04	1 00042 1723:	BICB2 MOVL RET	#8, 1(TOKEN) TOKEN, RO	
FF60	57	14 AE	2701 8F 000000000 EF 000000000 FF	F1	00D4A 194\$: 00D50	ACBL	#9, #1, INDEX, 189\$ #9985, TOKENBUFFER CHARPTR aCHARPTR, R1 CHARTBL[R1], R0 (R0), 196\$ #1, (R0), 196\$ #2, (R0), 197\$ R1, #13	6505 6546 6548 6549
		51	2701 8F 000000000 EF 000000000 FF 00000000 EF41	B0 94	00056 195\$:	MOVZBL	CHARPTR R1	: 6548
	04	50 60 60	60	DE E 8	0006B	BLBS	(RO), 196\$	6550
	19	60	02 51	E1	00D72 00D76 196\$:	BBC CMPB	#2. (RO). 197\$ R1. #13 197\$	6550 6551 6553
		20	14 AE	91	00D79 00D7B	BEQL CMPB	197\$ TOKENBUFFER, #32 197\$	
		50	14 AE 0E 14 AE 14 AE 51	96 94		INCL MOVZBL MOVAL BLBS BBS BBC CMPB BEQL CMPB BGEQU INCB MOVZBL	TOKENBUFFER RO	6555
		14 AE40	C7	90	00D88 00D8D	MOVB BRB PUSHAB	TOKENBUFFER TOKENBUFFER RO R1 TOKENBUFFER (RO) 195\$ TOKENBUFFER	6557
			14 AE 01	9F DD	00D8F 197\$:	PUSHAB		: 6560
	000	000000 00	00028D30 8F 03 000000000 EF 01 A0 06 02 A0 00 00028992 8F	DD FB	00094 0009A 000A1 198\$:	PUSHL PUSHL CALLS MOVL	#167216 #3, LIB\$SIGNAL	6567
		ÓĎ	01 A0	91 13	00DAS 00DAC	CMPB	#3, L!B\$SIGNAL CHARPTR, RO 1(RO), #13 199\$ 2(RO), #39	: 0,00
		27	02 A0	91	00DB2	BEQL CMPB BEQL PUSHL	2(RO), #39 200\$ #166290	
	000	000000G 00	01	FB	00DB4 199\$: 00DBA 00DC1 200\$:	PUSHL CALLS MOVB	#166290 #1, LIB\$SIGNAL	6569
15 AE	18 000	000000 E		FO	00DC5 00DCF 00DD6 201\$:	INSV ADDL2 PUSHAB	#1, LIB\$SIGNAL #3, TOKENBUFFER aCHARPTR, #0, #24, TOKENBUFFER+1 #3, CHARPTR TOKENBUFFER	6571 6572 6573 6574
			14 AE 01	9F	00DD6 201\$:	PUSHL		6574
	000	0000V CF		94 04	00009 00008 202\$: 000E0 000E1 203\$: 000E8 000EB	RET	#2, CREATE_OPERAND_TOKEN	4501
	000	000000. Et	14 AE	94 94	00DE1 203\$: 00DE8	MOVL CLRB CLRL	STARTPTR, CHARPTR TOKENBUFFER	6591 6592 6595
		00	00000000°FF40	91	00DED 204\$:	CMPB BEQL	aCHARPTR[I], #13	
		15 AE40	00000000°FF40	90	00DF5 00DF7	MOVB	aCHARPTR[1], TOKENBUFFER+1[1]	: 6596

DV

DBGPARSER V04-000	H 14 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 Page 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1								
	00000000G	50 00	14 14 000289E2	AE 14 AE 01 8F 050	96 00E01 F3 00E04 9F 00E08 2059 DD 00E0B DD 00E0D FB 00E13 D4 00E1A 04 00E1C	INCB AOBLEQ PUSHAB PUSHL PUSHL CALLS CLRL RET	TOKENBUFFER #20 I, 204\$ TOKENBUFFER #1 #166370 #3, LIB\$SIGNAL R0	6597 6593 6600 6601	

; Routine Size: 3613 bytes, Routine Base: DBG\$CODE + OA9E

DBG\$GL_CHARTBL = CHARTBL; CETBL = .PTR[0] + TABLEBASE;

CHSMOVE (256 * SUPVAL, BASE_CHARACTER_TABLE, CHARTBL);

Page 218 (24)

```
DBGPARSER
V04-000
                                                                                                                        16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                     VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
  INCR I FROM 0 TO .CETBL[-1] - 1 DO
                              6661
6662
6663
66665
66667
66670
6677
6678
6677
6678
6679
                                                           BEGIN

CEPTR = CETBL[.1];
CHARTBL[.CEPTR[CE_CHAR], CHRTBL$L_WHOLE_ENTRY] = .CEPTR[CE_BITS];
                                                        Set up the various parse table pointers to the tables for this language.
                                                   IDENT OPERATOR TABLE = .PTR[1] + TABLEBASE;

OPCHAR OPERATOR TABLE = .PTR[2] + TABLEBASE;

STATE TABLE = .PTR[3] + TABLEBASE;

PRIMARY TABLE = .PTR[4] + TABLEBASE;

SUBSCRIPT TERM TBL = .PTR[5] + TABLEBASE;

PRIDTBL = .PTR[6] + TABLEBASE;

BIF TABLE = .PTR[7] + TABLEBASE;

MULTIPLE SUBSCR = .PTR[8];

ENFORCE RECORD = .PTR[9];

CASING SIGNIFICANT = .PTR[10];

COMPONENTS IN PATHNAME = .PTR[11];

INCOMPLETE QUAL = .PTR[12];
                             6681
6682
6683
6684
6685
                                                     INCOMPLETE QUAL = .PTR[12]:
                                                        Initialize the Operator Evaluation tables and the Print tables for
                                                        the current language.
                                                    DBG$EVALOP_SET_LANGUAGE (.LANGUAGE);
DBG$PRINT_SET_LANGUAGE (.LANGUAGE);
                             6687
6688
                                                    RETURN:
                             6689
                                                    END:
                                                                                                                                                         DBG$PARSER_SET_LANGUAGE, Save R2,R3,R4,R5,-
R6,R7,R8,R9,R10
TABLEBASE, R10
CHARTBL, R9
LANGUAGE, R8
                                                                                                       07FC 00000
                                                                                                                                           .ENTRY
                                                                                                                                                                                                                                               6604
                                                                         5A
59
58
                                                                              00000000
                                                                                                                                           MOVAB
                                                                                                                00009
                                                                                                                                           MOVAB
                                                                                                          DO
19
                                                                                                                                          MOVL
                                                                                                                                                                                                                                               6645
                                                                                                                                          BLSS
                                                                                                                00014
                                                                                                          D1
14
                                                                                                                00016
                                                                         OA
                                                                                                                                           CMPL
                                                                                                                                                         R8, #10
                                                                                                                                                                                                                                               6646
                                                                                                                                          BGTR
                                                                                                          9E
C1
11
                                                                         50
                                                                                                                0001B
0001E
                                                                                     2F91 CA48
                                                                                                                                                         TABLEBASE, RO
LANGUAGE_TABLE_PTRS[R8], RO, PTR
                                                                                                                                          MOVAB
                                                                                                                                                                                                                                               6648
                                              56
                                                                                                                                          ADDL3
                                                                                                                00025
                                                                                                                                          BRB
                                                                                                                                                         TABLEBASE, RO
LANGUAGE_TABLE PTRS+40, RO, PTR
CHARTBL, DBG$GC_CHARTBL
TABLEBASE, RO
(PTR), RO, CETBL
                                                                                                                00027 18:
                                                                                                          9E
C1
9E
C1
28
                                                                                                                                          MOVAB
                                                                                                                                                                                                                                               6651
                                                                                      2FB9
                                                                                                   CA 69 6A 66 8F 01
                                                                                                                0002A
                                                                                                                                          ADDL3
                                                    00000000
                                                                                                                00030 25:
                                                                                                                                          MOVAB
                                                                                                                00037
                                                                                                                                          MOVAB
                                                                                                                                                                                                                                               6659
                                                                                                                0003A
                                                                                                                                          ADDL3
MOVC3
                                                            0591
                                                                                      0400
                                                                                                                0003E
                                                                                                                                                         #1024, BASE_CHARACTER_TABLE, CHARTBL
                                                                                                                                                                                                                                               6660
                                                                                                                00046
                                                                                                                                           MNEGL
                                                                                                                                                                                                                                               6661
                                                                                                   0E
                                                                                                                00049
                                                                                                                                          BRB
                                                                                                                                                         (CETBL)[1], CEPTR
3(CEPTR), RO
#0, #24, (CEPTR), CHARTBL[RO]
                                                                                                          DE
9A
                                                                                                                0004B 3$:
                                                                                                                                          MOVAL
                                                                                                                                                                                                                                               6663
                                                                                                                0004F
                                                                                                                                          MOVZBL
                                                                                                                                                                                                                                               6664
                6940
                                              62
                                                                                                                00053
                                                                                                                                          EXTZV
```

DBGPARSER V04-000 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRCJDBGPARSER.B32;1	Page 220 (24)
ED 51 FC A7 F2 00059 4\$: AOBLSS -4(CETBL), I 3\$ 040C C9 04 B640 9E 00061 MOVAB TABLEBASE, R0 0418 C9 08 B640 9E 00068 MOVAB TABLEBASE, R0 0418 C9 08 B640 9E 00068 MOVAB TABLEBASE, R0 0424 C9 0C B640 9E 00072 MOVAB TABLEBASE, R0 0410 C9 10 B640 9E 00075 MOVAB TABLEBASE, R0 0410 C9 10 B640 9E 00076 MOVAB TABLEBASE, R0 0410 C9 10 B640 9E 00076 MOVAB TABLEBASE, R0 0428 C9 14 B640 9E 00089 MOVAB TABLEBASE, R0 0428 C9 14 B640 9E 00089 MOVAB TABLEBASE, R0 0420 C9 18 B640 9E 00093 MOVAB TABLEBASE, R0 0420 C9 18 B640 9E 00093 MOVAB TABLEBASE, R0 0420 C9 18 B640 9E 00093 MOVAB TABLEBASE, R0 0420 C9 18 B640 9E 00093 MOVAB TABLEBASE, R0 0410 C9 20 A6 D0 0009A MOVAB TABLEBASE, R0 0414 C9 20 A6 D0 0009A MOVAB TABLEBASE, R0 0404 C9 24 A6 D0 0009A MOVAB TABLEBASE, R0 0404 C9 24 A6 D0 0000A MOVAB TABLEBASE, R0 0404 C9 24 A6 D0 0000A MOVAB TABLEBASE, R0 0404 C9 24 A6 D0 0000A MOVAB TABLEBASE, R0 0404 C9 24 A6 D0 0000A MOVAB TABLEBASE, R0 0404 C9 24 A6 D0 0000A MOVAB TABLEBASE, R0 0400 C9 24 A6 D0 0000A MOVAB TABLEBASE, R0 0400 C9 24 A6 D0 0000A MOVAB TABLEBASE, R0 0400 C9 24 A6 D0 0000A MOVAB TABLEBASE, R0 0400 C9 24 A6 D0 0000A MOVAB TABLEBASE, R0 0400 C9 24 A6 D0 0000A MOVAB TABLEBASE, R0 0410 C9 30 A6 D0 0000A MOVAB TABLEBASE, R0 0410 C9 30 A6 D0 0000B MOVAB TABLEBASE, R0 0410 C9 30 A6 D0 0000B MOVAB TABLEBASE, R0 0410 C9 30 A6 D0 0000B MOVAB TABLEBASE, R0 0410 C9 30 A6 D0 0000B MOVAB TABLEBASE, R0 0410 C9 30 A6 D0 0000B MOVAB TABLEBASE, R0 0410 C9 30 A6 D0 0000B MOVAB TABLEBASE, R0 0410 C9 30 A6 D0 0000B MOVAB TABLEBASE, R0 0411 FB 0000C CALLS #1, DBS\$PRINT_SET_LANGUAGE	6661 6670 6671 6672 6673 6674 6675 6676 6676 6679 6680 6681 6686

; Routine Size: 211 bytes, Routine Base: DBG\$CODE + 18BB

GLOBAL ROUTINE DBG\$PRIMARY PARSER (OPERAND EXPECTED FLAG, ADDRESS EXPRESSION, TERM_LIST, PAREN_NESTING, RET_TOKEN, RET_OPERAND_FLAG): NOVALUE =

FUNCTION

This routine parses Primary Symbols and serves as a get-token routine for the Expression Parser. It calls the Lexical Scanner to get lexical tokens from the command line being parsed. It then intercepts tokens which are part of a Primary Symbol and uses those to build a Primary Descriptor for the symbol. Lexical tokens which are not part of Primary Symbols are simply passed through to the Expression Parser. The result is that the Expression Parser sees a stream of operators and operands where each Primary Symbol or constant has been preparsed and packaged as a single operand by the Primary Parser.

A 'Primary Symbol' is defined to be a variable name which may include pathname qualification, subscripting, data component selection, and dereferencing. Exactly which of these are allowed depends on the current language. Thus 'X' and 'MOD\ROUT\Z' are Primary Symbols and so is 'M\R\X(2,3).Y^.Z(4)'. In effect, a Primary Symbol is anything that can be described by a Primary Descriptor (see DBGLIB.REQ).

The Primary Parser emulates a finite-State Machine (FSM) to parse the Primary Symbols accepted in the current language. The FMS for the current language is defined by a Primary Parser State Table which defines which operators (such as '\', '.', and subscripting) may appear in which order in a Primary Symbol. For each transition in the FMS, a semantic routine is executed which builds up the Primary Descriptor for the current Primary Symbol (or a Value Descriptor if the current symbol is a constant). The symbol is "accepted" by the parser if a transition is reached which returns the completed Primary Descriptor to the caller. If the symbol is not accepted by the FSM, a syntax error is signalled.

The Primary Parser is called by the Expression Parser. However, the Primary Parser will itself call the Expression Parser to pick up subscript expressions within Primary Symbols. Hence these two routines call each other recursively, and their data structures have been set up so that this recursion will work properly.

INPUTS

OPERAND_EXPECTED_FLAG - A flag which is set to TRUE if the caller expects to see an operand next. This flag is used to determine whether certain operators (such as "+") are prefix (if an operand is expected) or infix (if an operand is not expected) operators at the current point in the parsing of an expression.

ADDRESS_EXPRESSION - A flag set to TRUE if we are parsing a DEBUG Address Expression instead of a language expression. This affects the parsing of Address Expression operators such as '+'', '-'', '', and 'a' which are recognized by DEBUG rules, not language rules, in Address Expressions.

TERM_LIST - A vector of pointers to Terminator Lexical Token Entries for the Terminator Tokens which can terminate the expression being parsed. The vector must be in PLIT form (TERM_LIST[-1] gives the number of entries) and each pointer is expected to

6695

LOCAL

Page 222

Pointer to the current Token Entry

Pointer to a Value Descriptor

TYPEID (Type ID) for current symbol

LAST_OPERAND: REF TOKENSENTRY,

NUMERIC_PATHNAME, OPCODE, OPERAND_EXPECTED,

PATHDESC: PTH\$PATHNAME,
PATHSTRING,
PATHVECTOR: REF VECTOR[,LONG],
PLIPTR: REF DBG\$PRIMARY,
PRID: REF PRID\$ENTRY,
PRIMPTR: REF DBG\$PRIMARY,
SAVED_PATHDESC: PTH\$PATHNAME,
STATE_INDEX,

STATUS, STRDESC: BLOCK[8,BYTE], SUBSCR_DESC: SUBSCR\$DESC, SYMID, TEMPTOKEN: REF TOKEN\$ENTRY, TOKEN: REF TOKEN\$ENTRY, TYPEID, VALPTR: REF DBG\$VALDESC;

There are two different initialization paths. The normal path is when we are picking up a Primary from scratch; that is the ELSE clause below.

IF ACTUALCOUNT() GTR 6
THEN
BEGIN

If we got an Operator Token last time which was not part of the Primary Symbol we were building, then we saved it in SAVED_TOKEN while we completed and returned the Primary Descriptor. In that case, return the input Primary and retain the SAVED_TOKEN value for the next time Primary Parser is called.

IF .SAVED_TOKEN NEQ 0

BEGIN
RET_TOKEN[0] = ACTUALPARAMETER(7);
RETURN;
END:

! This is the case where we call the Primary Parser when we already! have constructed part of the Primary, and we want to pick up the ! rest of the Primary. In this case, a pointer to the partially-

DI

Page 224 (25) C 15 16-Sep-1984 02:10:13 14-Sep-1984 12:17:30

IF .DBG\$GL_DEVELOPER[2] THEN DUMP_TOKEN(.TOKEN);

Check for an invocation number. An invocation number consists of an integer constant in a pathname where an operator is expected. If this is an invocation number, we convert it to the invocation number postfix operator which then passes through the rest of the code below in the normal way.

IF (.PRIMPTR EQL 0)

(NOT .OPERAND EXPECTED)

(.LAST OPERAND NEQ 0)

(.TOKEN[TOKENSB_KIND] EQL TOKENSK_OPERAND)

.TOKEN_IS_INTEGER[.TOKEN[TOKENSW_CODE]] AND AND AND THEN

END:

Handle operands. If this is an operand, check that we are actually expecting an operand at this point. Save a pointer to the operand and loop to get the next token.

IF .TOKEN[TOKEN\$B_KIND] EQL TOKEN\$K_OPERAND THEN

> BEGIN IF NOT . OPERAND_EXPECTED THEN

SIGNAL (DBG\$_MISINVOPER, 1, TOKEN[TOKEN\$B_LENGTH]);

OPERAND EXPECTED = FALSE; LAST_OPERAND = .TOKEN;

Handle operators. If this operator is not part of the current Primary Symbol, we save it while building and returning a descriptor for the Primary Symbol. If the operator is part of the current Primary Symbol, we add to the Primary we are building and loop to pick up more of the Primary.

ELSE BEGIN

> If this operator is not part of the Primary Symbol we are building (if any), then it is a language or address expression opera-! tor. Save it for the next call on DBG\$PRIMARY_PARSER and use the

```
IF .TOKEN[TOKEN$B_KIND] NEQ TOKEN$K_POSTFIX_OP
THEN
OPERAND_EXPECTED = TRUE;
```

Get the Operator Code for this Primary Operator and loop through the transitions for the current state in the Primary Parser State Table until we find a transition for this operator. If we find no such transition (PRIMARYSB_OPCODE field zero), the current operator is not allowed in this context, so we signal a syntax error. If the transition is allowed, we pick up its action index and the next state in the FSM.

OPCODE = .TOKEN[TOKENSW_CODE];
WHILE .PRIMARY_TABLE[.STATE_INDEX, PRIMARYSB_OPCODE] NEQ .OPCODE DO
BEGIN
IF .PRIMARY_TABLE[.STATE_INDEX, PRIMARYSB_OPCODE] EQL 0
THEN
SIGNAL(DBG\$_SYNERREXPR, 1, TOKEN[TOKENSB_OPLEN]);

STATE_INDEX = .STATE_INDEX + 1; END;

ACTION = .PRIMARY TABLE[.STATE_INDEX, PRIMARY\$B_ACTION]; STATE_INDEX = .PRIMARY_TABLE[.STATE_INDEX, PRIMARY\$W_NEXTSTATE];

Execute the action routine associated with this state transition.

CASE _ACTION FROM PRIMARY\$K_MIN_ACTION TO PRIMARY\$K_MAX_ACTION OF SET

! Handle Global Symbol backslash operator (prefix "\"). Do ! nothing at this point. [PRIMARY\$K_ACT_START_GBL]: 0;

Handle terminator after Global Symbol backslash. Just pick up the global symbol name and create a Primary Descriptor for it. Then exit from the parse loop.

EPRIMARY\$K_ACT_GBL_TERM]:
 BEGIN
 PATHDESC[PTH\$B_TOTCNT] = 1;
 PATHDESC[PTH\$B_PATHCNT] = 1;
 PATHVECTOR[0] = UPLIT BYTE(0);
 APPEND_TO_PATHNAME(PATHDESC, LAST_OPERAND, NOT_REC_COMP);
 PRIMPTR = PATHNAME_TO_PRIMARY(PATHDESC, SUBSCR_DESC, PLIPTR, SAVED_PATHDESC);
 EXITLOOP;
 EXITLOOP;
 END;

! Check that an invocation number has not already been

Page 228 (25)

```
H 15
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
                                                                                                                                                                                                                                                                 VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGPARSER.B32;1
V04-000
    70998
70998
70998
71003
71008
71008
71008
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
71108
                                                                                                                                                        THEN
                                                                                                                                                                 BEGIN
IF .KIND EQL DEFINE_ADDRESS
OR .KIND EQL DEFINE_VALUE
                                                             BEGIN
                                                                                                                                                                                    We have found a matching DEFINEd symbol. Copy the descriptor into temporary memory.
                                                                                                                                                                                     (fourth parameter FALSE <-> copy into tempmem).
                                                                                                                                                                               DBG$NCOPY_DESC (.PRIMPTR, PRIMPTR,
                                                                                                                                                                                                                                 DUMMY, FALSE);
                                                                                                                                                                               LEAVE TEMP_BLOCK;
END;
                                                                                                                                                                   END;
                                                                                                                                                        END:
                                                                                                                                            END:
                                                                                                                                APPEND TO PATHNAME (PATHDESC, .LAST_OPERAND, NOT_REC_COMP);
PRIMPTR = PATHNAME_TO_PRIMARY (PATHDESC, SUBSCR_DESC,
                                                                                                                                                                                                                      .PLIPTR, SAVED_PATHDESC);
                                                                                                                                END; ! TEMP_BLOCK
                                                                                                                                END:
                                                                                                                          Handle a dot immediately after the start of the symbol.
                                                                                                                          In PLI we collect all the record components before
                                                                                                                          calling PATHNAME_TO_PRIMARY.
                                                                                                                     [PRIMARY$K_ACT_START_DOT_PLI]:
                                                                                                                                BEGIN
                                                                                                                                 APPEND_TO_PATHNAME(PATHDESC, .LAST_OPERAND, NOT_REC_COMP);
                                                                                                                          Handle a dot after the start of the symbol in COBOL.
                                                                                                                     [PRIMARY$K_ACT_START_DOT_COB]:
                                                                                                                                BEGIN
                                                                                                                                 APPEND_TO_PATHNAME(PATHDESC, .LAST_OPERAND, COB_REC_COMP);
                                                                                                                          Handle a subscript parenthesis immediately after the start of
                                                                                                                           the symbol. Append the last operand to the Pathname Descrip-
                                                                                                                          tor, build a partial Primary Descriptor, and then pick up the
                                                                                                                           subscript expressions within the parentheses.
                                                                                                                     [PRIMARY$K_ACT_START_SUBSCR]:
                                                                                                                                 BEGIN
                                                                                                                                LABEL TEMP_BLOCK;
                                                                                                                                      If the last operand was an identifier, we append it to
                                                                                                                                       the current Pathname Descriptor and convert that to a
                                                                                                                                  ! Primary Descriptor.
```

Page 231 (25)

! If the last operand was not an identifier then ! it is not legal to select a component.

```
DBGPARSER
V04-000
                                                                                                             VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1
                                                    built-in function name. This will return a primary that represents the value of the built-in function.
[PRIMARY$K_ACT_START_BIF_CALL]:
                                                      BEGIN
                                                        If the last operand was not a built-in function name,
                                                         signal the error.
                                                          .LAST_OPERAND[TOKEN$W_CODE] NEQ TOKEN$K_BUILTIN_FUNCTION
                                                           SIGNAL (DBG$_MISOPEMIS, 1, TOKEN[TOKEN$B_OPLEN]);
                                                       ! Call a routine that returns a counted longword vector of
                                                         pointers that point to primary tokens of the arguments
                                                         of the specified built-in function call. The number of
                                                         arguments expected in the B_FLAGS field of a built-in
                                                         function operand.
                                                      ARG_LIST = DBG$GET_BIF_ARGUMENTS(.LAST_OPERAND[TOKEN$B_FLAGS]
                                                                                               LAST_OPERAND[TOKENSB_[ENGTH]);
                                                        for now, we will not accept any built-in function that contains more than 2 arguments. If more than 2 are present, an invalid argument list message in signaled.
                                                       IF .ARG_LIST[0] LEQ 2
                                                       THEN
                                                           BEGIN
                                                             Create a new operator token to represent the specific built-in function call and then pass this token to
                                                              DBG$EVAL_LANG_OPERATOR to proccess the built-in
                                                              function and return its value as a primary.
                                                            IF .ARG_LIST[0] EQL 1
                                                                TOKEN = CREATE_OPERATOR_TOKEN(.LAST_OPERAND[TOKEN$B_BIF],
LAST_OPERAND[TOKEN$B_EENGTH],
                                                                                                      .TOKEN[TOKEN$B_KIND]7
                                                           ELSE
                                                                TOKEN = CREATE_OPERATOR_TOKEN(.LAST_OPERAND[TOKEN$B_BIF],
LAST_OPERAND[TOKEN$B_ENGTH],
TOKEN$K_INFIX_OP);
                                                           PRIMPTR = DBG$EVAL_LANG_OPERATOR(.TOKEN, .ARG_LIST[1], .ARG_LIST[2]);
                                                             Return the fact that we have built the result by now
                                                              and that it is a value descriptor. Because of the
                                                              way the BIF tokens come through the above initialization
                                                             code, this value was never set. And It Better Be!
                                                           RET_OPERAND_FLAG[0] = TRUE;
                                                           END
                                                      ELSE
```

```
N 15
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                             VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1
                                                            SIGNAL (DBG$_INVARGLIS, 1, LAST_OPERAND[TOKEN$B_LENGTH]);
                                                      EXITLOOP;
                                                      END:
  Handle the Ada tick operator immediately after the start of the symbol. The symbol thus consists of just a single type
                                                 [PRIMARY$K_ACT_START_TICK]:
BEGIN
                                                        If the last operand was an identifier, we append it to the current Pathname Descriptor and convert that to a
                                                         Primary Descriptor.
                                                      IF .LAST_OPERAND[TOKEN$W_CODE] EQL TOKEN$K_IDENTIFIER
                                                      THEN
                                                           BEGIN
                                                             First check for DEFINEd symbols.
                                                             Check that no invocation number is present.
                                                           IF .PATHDESC[PTH$B_LOCINVOC] EQL 0
                                                           THEN
                                                                  Look up the symbol in the DEFINE symbol table.
                                                                IF DBG$DEF_SYM_FIND (LAST_OPERAND [TOKEN$B_LENGTH],
                                                                                          KIND, PRIMPTR,
                                                                                          DUMMY, DUMMY)
                                                                THEN
                                                                     BEGIN
                                                                     IF .KIND EQL DEFINE ADDRESS OR .KIND EQL DEFINE VALUE
                                                                     THEN
                                                                          BEGIN
                                                                            We have found a matching DEFINEd symbol.
                                                                            Copy the descriptor into temporary memory.
                                                                            (fourth parameter FALSE <-> copy into tempmem).
                                                                          DBGSNCOPY_DESC (.PRIMPTR, PRIMPTR,
                                                                                               DUMMY, FALSE);
                                                                          EXITLOOP;
                                                                          END:
                                                                     END:
                                                                END:
                                                           APPEND_TO_PATHNAME(PATHDESC, .LAST_OPERAND, NOT_REC_COMP);
```

```
Call GETSYMBOL directly since all we need is the typeid. TRUE is passed in to tell GETSYMBOL that we do want it
           to look up symbol-types as well as the normal data-types.
        DBG$STA_GETSYMBOL (PATHDESC, TYPEID, KIND, 0, 0, 0, TRUE);
           Check the output from getsymbol to see if a unique symbol
           was found. If not, signal the appropriate error.
         IF .TYPEID EQL O
         THEN
            DBG$NPATHDESC_TO_CS(PATHDESC, PATHSTRING);
IF .KIND EQL RST$K_NOTUNIQUE
THEN
                  SIGNAL (DBGS_NOUNIQUE. 1, .PATHSTRING)
             ELSE
                    .KIND EQL RSTSK_OVERLOAD
                      SIGNAL (DBG$_NOTUNGOVR, 1, .PATHSTRING)
                 ELSE
                      SIGNAL (DBG$_NOSYMBOL, 1, .PATHSTRING);
             END:
        PRIMPTR = DBG$EVAL_ADA_TICK(.TYPEID, .TOKEN);
           Return the fact that we have built the result by now
           and that it is a value descriptor. Because of the
           way the Ada tick tokens come through the above initialization code, this value was never set.
           And It Better Be!
        RET_OPERAND_FLAG[0] = TRUE;
    ELSE
        SIGNAL (DBG$_SYNERREXPR, 1, .LAST_OPERAND[TOKEN$B_LENGTH]);
    EXITLOOP:
    END:
 Handle the terminator operator immediately after the start of
 the symbol. The symbol thus consists of just a single name
 or a constant.
[PRIMARY$K_ACT_START_TERM]:
    BEGIN
      If the last operand was an identifier, we append it to
      the current Pathname Descriptor and convert that to a
      Primary Descriptor.
    IF .LAST_OPERAND[TOKENSW_CODE] EQL TOKENSK_IDENTIFIER
```

END:

! Handle the backslash operator after a previous backslash ope-

Handle a subscript after a backslash in language PLI. In PLI we do not incorporate the subscripts into the Primary Descriptor until later.

Page 239 (25)

[PRIMARY\$K_ACT_SLASH_SUBSCR_PLI]: BEGIN

7664

LOCAL

NAME:

Page 240 (25)

Page 241 (25)

Handle the terminator operator after a backslash. Here we just complete the Pathname Descriptor and convert it to a Primary Descriptor. We are then done with the symbol.

[PRIMARYSK_ACT_SLASH_TERM]:

BEGIN

APPEND_TO_PATHNAME(PATHDESC, .LAST_OPERAND, NOT_REC_COMP);

PRIMPTR = PATHNAME_TO_PRIMARY(PATHDESC, SUBSCR_DESC, .PLIPTR, SAVED_PATHDESC);

EXITLOOP; END;

END:

Handle a slash after an "OF" in COBOL.

[PRIMARY\$K_ACT_DOT_SLASH_COB]: APPEND_TO_PATHNAME(PATHDESC, .LAST_OPERAND, NOT_REC_COMP);

Handle a dot (data qualification) after another dot. Check that the last operand is an identifier and that it is a valid component name of the current record type. Get its SYMID, etc., and add it to the Primary Descriptor being built.

[PRIMARY\$K_ACT_DOT_DOT]: GET_RECORD_COMPONENT(.PRIMPTR, LAST_OPERAND[TOKEN\$B_LENGTH]);

Handle a dot after another dot in PLI. In PLI we do not call PATHNAME_TO_PRIMARY until after collecting all the record components.

[PRIMARY\$K_ACT_DOT_DOT_PLI]: BEGIN APPEND_TO_PATHNAME (PATHDESC, .LAST_OPERAND, REC_COMP);

Handle an 'Of' operator after another 'Of' operator COBOL.

[PRIMARY\$K_ACT_DOT_DOT_COB]: BEGIN APPEND_TO_PATHNAME (PATHDESC, .LAST_OPERAND, COB_REC_COMP);

Handle subscripting after a dot (data qualification). Check that the last operand is an identifier and that it is a valid component name of the current record type. Get its SYMID, etc., and add it to the Primary Descriptor being built. Then pick up all the subscript expressions and add their values to the Primary Descriptor.

```
H 16
DBGPARSER
V04-000
                                                                                              16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                  VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32;1
                                                           [PRIMARYSK_ACT_DOT_SUBSCR]:
    BEGIN
    GET_RECORD_COMPONENT(.PRIMPTR, LAST_OPERAND[TOKENSB_LENGTH]);
    GET_SUBSCRIPTS(.PRIMPTR);
    END;
                       Handle a subscript after a dot in PLI. In PLI we do not
                                                              call PATHNAME_TO_PRIMARY until after picking up all
                                                              the record components.
                                                           [PRIMARY$K_ACT_DOT_SUBSCR_PLI]:
                                                                 BEGIN
                                                                 APPEND TO PATHNAME (PATHDESC, LAST OPERAND, REC_COMP); SAVE_SUBSCRIPTS (PATHDESC, SUBSCR_DESC);
                                                                 END:
                                                              Handle the subscript operator after the "Of" operator
                                                              in COBOL.
                                                           [PRIMARY$K_ACT_DOT_SUBSCR_COB]:
                                                                 BEGIN
                                                                 APPEND TO PATHNAME (PATHDESC, LAST OPERAND, NOT_REC_COMP);
SAVE_SUBSCRIPTS (PATHDESC, SUBSCR_DESC);
                                                                 END:
                                                             Handle the PASCAL dereference operator (*) occurring after a dot (data qualification). Here we call a routine which
                                                             appends the last operand (representing a record component) onto the Primary Descriptor being built. We then call the routine that handles derefencing—it just lights the EVAL bit on the sub-node and then allocates a new subnode for
                                                              the object being pointed to.
                                                           [PRIMARY$K_ACT_DOT_DEREF]:
                                                                 BEGIN
                                                                 GET_RECORD_COMPONENT(.PRIMPTR, LAST_OPERAND[TOKEN$B_LENGTH]);
GET_DEREFERENCE(.PRIMPTR);
                                                                 END:
                                                              Handle a PLI dereference as in A.B->C. Here we convert the
                                                              left-hand-side to a Primary, and then save away the Primary.
                                                              We loop back to the start state to pick up the rest
                                                              of the expression.
                                                           [PRIMARYSK_ACT_DOT_DEREF_PLI]:
                                                                 BEGIN
                                                                   Save away the Primary constructed so far.
                                                                 APPEND_TO_PATHNAME (PATHDESC, .LAST_OPERAND, REC_COMP);
PLIPTR = PATHNAME_TO_PRIMARY (PATHDESC, SUBSCR_DESC, .PLIPTR, SAVED_PATHDESC);
                                                                 IF (.PLIPTR[DBG$B_DHDR_KIND] NEQ RST$K_DATA) OR
                                                                      (.PLIPTR[DBG$B_DHDR_FCQDE] NEQ RST$R_TYPE_PTR)
                        8001
```

Page 243

Page 244 (25)

[PRIMARYSK_ACT_SUBSCR_DOT_PLI]:

LOCAL

DBGSNPATHDESC_TO_CS(PATHDESC, NAME);

Page 247 (25)

VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.832:1

```
M 16
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                    VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
                                                                                                                                                                   Page 248
(25)
                                                       cates a new subnode for the object being pointed to.
 812234567890123345678901234567890123
12234567890123345678901234567890123
                                                    LPRIMARY$K ACT DEREF DEREF]:
    GET_DEREFERENCE(.PRIMPTR);
                                                       Handle a PASCAL dereference operator (*) followed by a termi-
                                                       nator. The Primary Descriptor is now complete, so we exit
                                                       the parse loop.
                                                    [PRIMARY$K_ACT_DEREF_TERM]:
EXITLOOP;
                                                       Any other CASE index constitutes an internal DEBUG error.
                                                    [INRANGE, OUTRANGE]:
                                                         $DBG_ERROR('DBGPARSER\PRIMARY_PARSER 10');
                                                    TES:
                                               END:
                                                                                    ! End of ELSE-clause for operators
                                          END:
                                                                                    ! End of the get-token loop
                                       We are all done parsing the primary. Return a pointer to the
                                       descriptor we have constructed.
                                    RET_TOKEN[0] = .PRIMPTR;
RETURN;
                                    END:
                                                                                                 .PSECT DBG$PLIT, NOWRT, SHR, PIC, O
                                                                              031A0 TOKEN_IS_INTEGER:
BYTE 4
031A2 BLKB 1
                                                                         00
08
10
65
                                                                              031A4
031A5
031AE
031BD
                                                                                      P.AXH:
P.AXI:
                                                                                                 .BYTE
                                                                                                           <8>\decimal \
<16>\longword integer\
                                                                    6467442
                                                                                                 .ASCII
                                                                                      P.AXJ:
                                                                         1B
                                                                              031BF
                                                                                      P.AXK:
                                                                                                 .ASCII
                                                                                                           <27>\DBGPARSER\<92>\PRIMARY_PARSER 10\
                                                                                                 .PSECT
                                                                                                           DBG$CODE, NOWRT, SHR, PIC, O
                                                                        OFFC 00000
                                                                                                 .ENTRY
                                                                                                           DBGSPRIMARY_PARSER, Save R2,R3,R4,R5,R6,R7,-: 6691
                                                                                                           R8, R9, R10, RT1
-1112(SP), SP
SAVED_TOKEN, RO
                                                   5E FBA8
                                                                                                 MOVAB
                                                                                                 MOVL
                                                                                                                                                                        6851
```

DBGPARSER VO4-000			B 1 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 F 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 249 (25)
	· ·	06 6C	91 0000E CMPB (AP), #6 1B 00011 BLEQU 2\$ D5 00013 TSTL RO	: 6840
	14	50 06 BC 1C AC	13 00015 BEQL 1\$	6851
		56 01	04 0001C RET CE 0001D 15: MNEGL #1, LAST OPERAND	; 6853 ; 6867
	10 24	56 01 AE 04 AC 59 20 AC AE 1C AC 39	DO 00020 MOVL OPERAND_EXPECTED_FLAG, OPERAND_EXPECTED DO 00025 MOVL 32(AP), STATE_INDEX DO 00029 MOVL 28(AP), PRIMPTR	6854 6853 6867 6868 6869 6870 6840 6882
		39 50	11 0002E BRB 4\$	
		58 00000000 EF	DO 00034 MOVL RO, TOKEN D4 00037 CLRL SAVED_TOKEN	6885
	10	00D1 56 AE 04 AC 59	DO 00042 MOVL OPERAND_EXPECTED_FLAG, OPERAND_EXPECTED	6885 6886 6887 6897 6898 6899 6900
		24 AE 08 AE 00	D4 00047 CLRL STATE_INDEX D4 00049 CLRL PRIMPTR D4 0004C CLRL NUMERIC_PATHNAME	6900
00D0 8F		FF30 CD	00056	6902
0270 8F	00	5B FF38 CD 6E 00 40 AE 57	00065	: 6904
		7E OC AC 08 AC 1C AE	D4 00067 CLRL PLIPTR 7D 00069 48: MOVQ TERM_LIST, -(SP) DD 0006D PUSHL ADDRESS_EXPRESSION	6905
	F098	CF 04	DD 00070 PUSHL OPERAND_EXPECTED FB 00073 CALLS #4, DBG\$LEXICAL_SCANNER	6918
		58 50 00 02 58	DO 00083 PUSHI TOKEN	6920
	0000v	CF 01 5A 24 AE 55	FB 00085	6929
		51 10 AE	DO 0008A 5\$: MOVL PRIMPTR, R10 12 0008E BNEQ 6\$ E8 00090 BLBS OPERAND EXPECTED, 6\$ D5 00094 TSTL LAST_OPERAND 13 00096 BEQL 6\$ 91 00098 CMPB (TOKEN), #1 B12 0009B BNEQ 6\$ 3C 0009D MOVZWL 2(TOKEN), R0	6930
		01 68 48	13 00096 BEQL 6\$ 91 00098 CMPB (TOKEN), #1 3 12 0009B BNEQ 6\$	6932
	3c 00000000°	50 02 A8 EF 50	3C 0009D MOVZWL 2(TOKEN), RO E1 000A1 BBC RO, TOKEN_IS_INTEGER, 6\$	6933
		50 08 A8 50 04 50 04	3C 0009D MOVZWL 2(TOKEN), RO E1 000A1 BBC RO, TOKEN_IS_INTEGER, 6\$ 9A 000A9 MOVZBL 8(TOKEN), RO C0 000AD ADDL2 #4, RO C6 000B0 DIVL2 #4, RO O 9F 000B3 PUSHAB 3(RO)	
		03 AO	DO OOORD MOVE RO. TEMPTOKEN	6936
	04 04 04	00 01 AE 50 BE 04 BE 0100 8F 10 09 50 08 A8	0 00 000BD MOVL RO, TEMPTOKEN 90 000C1 MOVB #4, aTEMPTOKEN 8 A8 000C5 BISW2 #256, aTEMPTOKEN 9 FO 000CB INSV #9, #16, #16, aTEMPTOKEN 8 9A 000D1 MOVZBL 8(TOKEN), RO	6938 6939 6940 6941
04 BE	10	10 50 08 A8 50	9A 000D1 MOVZBL 8(TOKEN), RO	
	7E 04 9E 08	AE ÓC A8 50	C1 000D7 ADDL3 #12, TEMPTOKEN, -(SP) 28 000DC MOVC3 RO, 8(TOKEN), a(SP)+	6942

					C 1 16-Sep-19 14-Sep-19	084 02:10 084 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 250 (25)
	58	04	AE 68	00 000 91 000	E1 68:	MOVL	TEMPTOKEN, TOKEN	: 6943 : 6951
	12	10 08	1F AE A8 01 8F 03	12 000 E8 000 9F 000	DEA DEA DEE	BNEQ BLBS PUSHAB PUSHL	OPERAND_EXPECTED, 7\$ 8(TOKEN) #1	6954 6956
0000000G	00	000289AA 10	8F 03 AE 58	04 001	06 7\$:	PUSHL CALLS CLRL	#166314 #3, LIB\$SIGNAL OPERAND_EXPECTED	6958 6959
	56 1E	01	FF60 A8 56	51 001 E8 001	06 09 8\$:	MOVL BRW BLBS TSTL	TOKEN, CAST_OPERAND 48 1(TOK), 118 LAST_OPERAND	6959 6951 6979 6986
14	ВС	18	08 58 BC	DO 001	15	BNEQ	TOKEN, DRET_TOKEN DRET_OPERAND_FLAG	6989
000000001	EF 58	00000000	58 EF 01	04 001 00 001 9E 001 00 001	19 106.	CLRL RET MOVL MOVAB MOVL	TOKEN, SAVED_TOKEN PRIMARY_TERM_TOKEN, TOKEN #1, @RET_OPERAND_FLAG OPERAND_EXPECTED, 12\$	6988 6998 6999 7000
	99 02	10	AE 68 09	E9 001 91 001 12 001 E8 001 91 001	20 27 28 11\$: 2F 32	BLBC CMPB BNEQ	OPÉRAND EXPECTED, 12\$ (TOKEN), #2 13\$	7010
	17	10	AE 68 12 A8	12 001	30 129:	BLBS CMPB BNEQ	OPERAND_EXPECTED, 14\$ (TOKEN), #2 14\$	7012 7013
		0C 000289B2	01	9F 001 DD 001 DD 001	3D 13\$: 40 42 48 4F 14\$:	PUSHAB PUSHL PUSHL	12(TOKEN) #1 #166322	7015
0000000G	00		8F 03 68 24	13 001	48 4F 14\$:	CALLS CMPB BEQL	#3, LIB\$SIGNAL (TOKEN), #2	7018
FFFFFFF	8F		24 56 0E 56	13 001 05 001 13 001 01 001 13 001	54 56 58	TSTL BEQL CMPL	LAST_OPERAND	7020
	01		17	13 001 91 001 13 001	61	BEQL CMPB BEQL	LAST_OPERAND, #-1 16\$ (LAST_OPERAND), #1 16\$	7028
		0C 000289B2	66 12 A8 01 8F 03	9F 001	66 15\$:	PUSHAB PUSHL PUSHL	12(TOKEN)	7030
0000000G	00	00028782	03 68	FB 001	78 16\$:	CALLS	#166322 #3, LIB\$SIGNAL (TOKEN), #4 17\$	7035
10	AE AE 08	00000000	01 A8 FF49	DO 001 3C 001 DF 001	7D 81 17\$: 86 18\$:	BEQL MOVL MOVZWL PUSHAL CMPZV	#1, OPERAND_EXPECTED 2(TOKEN), OPCODE aprimary_table[state_index]	7037 7048 7049
	08	00000000	9F	ED 001 13 001 DF 001 95 001 12 001	95 90	BEQL PUSHAL TSTB	aprimary_table[state_index] a(sp)+	7051
		0C 000289E2	12 A8 01 8F	12 001 9F 001 DD 001 DD 001	A0 A3	BNEQ PUSHAB PUSHL	19\$ 12(TOKEN) #1	7053
0000000G	00	00020762	03 59	FB 001	AB B2 19\$:	PUSHL CALLS INCL	#166370 #3, LIB\$SIGNAL STATE_INDEX	7055

14 AE

DBGPARSER V04-000				D 1 16-Sep-1 14-Sep-1	984 02:10 984 12:17		Page 251 (25)
V04-000 18 AE. 59 0167 0169 0402 047B 04CB 0569 058B 0636 669 0636 669 0352	9E 9E 31 008C 0180 029D 00F3 057D 05AE 0448 06CA 06F5	00000000 00000000 01 01 0254 0462 0462 0440 0524 0579 059E 063A 06E2 06C1 06EB 0600	FF 409 FF 100000000000000000000000000000000000	14-Sep-1 11 001B4 DF 001B6 DF 001C3 EF 001CA CF 001CF 001D4 001EC 001FC 00204 0020C 00214 00224 00224 00224 00234	984 12:17 BRB PUSHAL EXTZV PUSHAL EXTZV CASEL .WORD	188 aPRIMARY TABLECSTATE INDEXI #8, #8, #3 (SP) + ACTION aPRIMARY TABLECSTATE INDEXI #16, #16, #16, #19, *STATE_INDEX ACTION, #1, #49 48-21s, - 22s-21s, - 23s-21s, - 31s-21s, - 84s-21s, - 94s-21s, - 95s-21s, - 97s-21s, -	7049 7058 7059 7064
		00000000	' EF	9F 00238	PUSHAB	55\$-21\$,- 58\$-21\$,- 98\$-21\$ P.AXK	8247

DV

FF32

					1	6-Sep-19 4-Sep-19	84 02:10 84 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRCJDBGPARSER.B32;1	Page 252 (25)
	0000000G	00	00028362	01 DI 8F DI 03 FI 76 1	0 0023E 0 00246 0 00246 0 00256 0 00266 0 00266 0 00266 0 00266		PUSHL PUSHL CALLS	#1 #164706 #3, LIB\$SIGNAL	
	FF30	CD 6B	000000000	8F B(EF 91 3BC 3	0024F 00256	22\$:	BRB MOVW MOVAB BRW	#3, LIB\$SIGNAL 26\$ #257, PATHDESC P.AXG, (PATHVECTOR) 69\$	7081 7083
0	3 00000000	50 EF	02	A6 30	00264	23\$:	MOVZWL BBS	3/I ACT ODEDAND) DO	7084
	FE5A FE58 FE5C	CD	010E 08 09 FF34 FE58	8F 99 3BC 33 4CE 86 4CE 86 4CD 99 4CD 99	0 0026F B 00276 E 00276 F 00282 F 00286	24\$:	MOVW MOVZBW MOVAB PUSHAB PUSHAB	RO, TOREN_IS_INTEGER, 24\$ 84\$ #270, STRDESC+2 8(LAST_OPERAND), STRDESC 9(R6), STRDESC+4 PATHDESC+4 STRDESC #2, OTS\$CVT_TI_L RO, STATUS STATUS, 25\$ STATUS 8(LAST_OPERAND)	7109 7111 7112 7113
	0000000G	00 AE 15	0C 0C 08	02 FI 50 DI AE EI AE DI A6 9	00287 000291 000295 000295 000296		MOVW MOVZBW MOVAB PUSHAB PUSHAB CALLS MOVL BLBS PUSHL PUSHAB PUSHL	O'EUSI O'EUUIN	7114
	00000000G FF30 FF31	OO CD CD 6B AE	0002898A 00000000°	A6 99 91 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	8 00276 00286 00286 00286 00286 00291 00296	25\$:	CALLS MOVB MOVW MOVAB	#166282 #4, LIB\$SIGNAL #1, PATHDESC #257, PATHDESC+1	7120 7121 7122 7123 7106 7149
	08	AE	FF32	72 1 CD 9	0 002C1 1 002C5 5 002C7 3 002CE	26\$: 27\$:	MOVL BRB TSTB BEQL	30\$	
	00000000G FE5A FE58 FE5C	00 CD CD	010E 00 00 00 FF34 FE58	CD 91	002C5 002C5 002C6 002C6 002C6 002C6 002C6	28\$:	PUSHL CALLS MOVW MOVZBW MOVAB PUSHAB PUSHAB	PATHDESC+2 28\$ #166418 #1, LIB\$SIGNAL #270, STRDESC+2 12(TOKEN), STRDESC 13(R8), STRDESC+4 PATHDESC+4 STRDESC #2, OTS\$CVT_TI_L R0, STATUS STATUS, 29\$ STATUS P.AXJ STRDESC P.AXI	7151 7157 7159 7160 7161
	000000006	OO AE 2D	00000000°	O2 FE 50 DO AE EE AE DO EF 90	B 002F5 0 002F6 B 00306 0 00304 F 00307)	CALLS MOVL BLBS PUSHL PUSHAB PUSHL PUSHAB PUSHL PUSHAB	#2, OTS\$CVT_TI_L R0, STATUS STATUS, 29\$ STATUS P.AXJ STRDESC	7162 7167 7166
			00000000° 00028E78 0C	A8 9	00317		PUSHAB PUSHL PUSHL PUSHAB PUSHL	P.AXI #3 #167544 12(TOKEN)	7165 7164
e c	00000000G FF30	00	0002898A	01 DI 8F DI 09 FI 01 8	1 00339	505:	PUSHL PUSHL CALLS ADDB3 BRB	#166282 #9, LIB\$SIGNAL #1, PATHDESC, PATHDESC+2	7169 7064
		01	02 FF 32	A6 B 1F 1 CD 9 19 1 AE 9	1 0033E 2 0033E 5 00345 2 00345 F 00347	31\$:	BRB CMPW BNEQ TSTB	2(LAST_OPERAND), #1 32\$ PATHDESC+2	7186
			20	AE 9	00347		PUSHAB	32\$ DUMMY	7201

		16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 253 (25)
00 03 01	24 A 2C A 3C A 08 A 05 02E 30 A	AE 9F 0034A PUSHAB DUMMY AE 9F 0034D PUSHAB PRIMPTR AE 9F 00350 PUSHAB KIND A6 9F 00353 PUSHAB 8(LAST_OPERAND) CALLS #5, DBG\$DEF_SYM_FIND CALLS #5, DBG\$DEF_SYM_FIND CALLS #5, DBG\$DEF_SYM_FIND CALLS #6, DBG\$DEF_	7206
	30 A 7 24 A 2C A 30 A	AE D1 00369 CMPL KIND, #5 E1 12 0036D BNEQ 32\$ TE D4 0036F 34\$: CLRL -(SP) AE 9F 00371 PUSHAB DUMMY AE 9F 00374 PUSHAB PRIMPTR AE DD 00377 PUSHL PRIMPTR	7207
00	02 1	CALLS #4, DBG\$NCOPY_DESC 15 31 00381 35\$: BRW 4\$ 16 B1 00384 36\$: CMPW 2(LAST_OPERAND), #1 17 12 00388 BNEQ 37\$ 18 PATHDESC+2 19 12 0038E BNEQ 37\$	7218 7263 7271
00 03 01 05	30 A	AE 9F 00390 PUSHAB DUMMY AE 9F 00393 PUSHAB DUMMY AE 9F 00396 PUSHAB PRIMPTR AE 9F 00399 PUSHAB KIND A6 9F 0039C PUSHAB &(LAST_OPERAND) A6 9F 0039F CALLS #5, DBG\$DEF_SYM_FIND B1 003A6 BLBS R0, 38\$ B1 003A9 37\$: BRW 75\$ B1 01 003AC 38\$: CMPL KIND, #1 B1 01 003BC CMPL KIND, #5 B1 12 003BC CMPL KIND, #5 B1 12 003BC CMPL KIND, #5	7278 7283 7284
00 01	30 A 00 02 A	AE 9F 003BA PUSHAB DUMMY AE 9F 003BD PUSHAB PRIMPTR AE DD 003CO PUSHL PRIMPTR 04 FB 003C3 CALLS #4, DBG\$NCOPY DESC	7305 7333
00 33 01 05	20 A 24 A 20 A 30 A 30 A 30 A 24 A	12 003D7	7341 7348 7353 7354 7363
	01 05 00 01 00 03 01 05	00 01 05 30 00 01 02 00 01 02 03 01 03 01 03 01 05 30 00 03 01 05 30 08 08 09 09 00 01 02 03 04 24 26 36 08 08 08 08 08 09 09 09 09 09 09 09 09 09 09	24 AE 9F 00340 PUSHAB PUSHAB BLANT OF RAND PUSHAB P

					15	5 1 5-Sep-19 4-Sep-19	84 02:10 84 12:17	:13 VAX-11 BLiss-32 V4.0-742 :30 EDEBUG.SRCJDBGPARSER.B32:1	Page 254 (25)
0000000G	00	30	AE 04	DD	00406		PUSHL	PRIMPTR #4. DBG\$NCOPY_DESC 78\$	1
		08	02AF A6	95	00413	42\$:	PUSHAB	8(LAST_OPERAND)	7365
000000006	00	000281A8	01 8F 03	DD	00416 00416 00418 00418 00425		PUSHL PUSHL CALLS	#164264	
	01	02	0277 A6	81	00428	438:	BRW CMPW	#3, LIB\$SIGNAL 77\$ 2(LAST_OPERAND), #1	7382
		FF32	1F CD 19	12	0042C 0042E 00432		BNEQ	PATHDESC+2	7415
		20	AE	9F	00432		BNEQ PUSHAB PUSHAB PUSHAB PUSHAB	45\$ DUMMY	: 7422
		20 24 20 30 08	AE	9F	0043A		PUSHAB	DUMMY PRIMPTR KIND	
000000006	00	őš	AE AE AE A6 O5 O	9F FB	00434 00437 0043A 0043D 00443		PUSHAB	8(LAST_OPERAND)	
	00		027F	E8	UU44D	45\$:	BLBS	#5, DBG\$DEF_SYM_FIND RO, 46\$ 80\$	
	01	30	AE 06 AE	D1 13	00450	46\$:	CMPL BEQL CMPL	KIND, #1	7427
	05	30	F1	12	00456 0045A		BNEQ	KIND, #5	7428
		24	AE	9F 9F	0045C 0045E	4/5:	BNEQ CLRL PUSHAB PUSHAB	-(SP) DUMMY	7437
000000006	00	24 20 30	7E AE AE O4	DD	00461 00464 00467 0046E 00471		PUSHL	PRIMPTR PRIMPTR #4, DBG\$NCOPY_DESC	
00000000	00		0281 7E 56 CD 03	DD FB 31 DD 9FB	0046E	48\$:	BRW	#4, DBG\$NCOPY_DESC 81\$ -(SP)	7449
		FF30	56 CD	DD 9F	00475		PUSHL	PATHDESC PATHDESC	1405
0000v	CF	FE60	03 CD	9F	00479		PUSHAB	#3, APPEND TO PATHNAME SAVED PATHDESC	7466
		48 FF30	AE	DD 9F	00482		PUSHAB	DITOTO	7467
0000v	CF 57	1130	04	FB	0048B		PUSHAB	SUBSCR_DESC PATHDESC #4, PATHNAME_TO_PRIMARY RO, PLIPTR 7(PLIPTR), #6 49\$ 6(PLIPTR), #16	
	06	07	A7	91	00490		CMPB	7(PLIPTR), #6	7468
	10	06	57 AE CO4 507 09 A7 03 03 AE CO2 AE	D9FFB012121FFBD113F	00482 00484 00487 0048B 00490 00497 00497 0049P		CALLS MOVL CMPB BNEQ CMPB BNEQ BRW	6(PLIPTR), #16	7469
		10	03D3	31 9F	0049F 004A2	498:	LO2HVR	49\$ 112\$ NAME	7474
000000006	00	FF30	02	9F FB	004A2 004A5 004A9 004B0 004B3		PUSHAB CALLS PUSHL	PATHDESC #2, DBG\$NPATHDESC_TO_CS	
		10	03B0	31	004B0 004B3		PUSHL BRW CMPW	NAME 111\$	7475
	10	02	12	13	004BA	50\$:	BEQL PUSHAB	2(LAST_OPERAND), #16	7499
		0C 000289B2	01	DD	004BA 004BC 004BF 004C1		PUSHL	12(TOKEN) #1	7501
0000000G	00	08	A6 12 A8 01 8F 03 A6	DD DD FB 9F	004C7 004CE	51\$:	PUSHL CALLS PUSHAB	#166322 #3, LIB\$SIGNAL 8(LAST_OPERAND)	7510

						1	5-Sep-19 4-Sep-19	84 02:10 84 12:17	VAX-11 Bliss-32 V4.0-742 CDEBUG.SRCJDBGPARSER.B32;1	Page 255 (25)
	EBAA	7E CF	01	A0585E58236630824383	9A FB DO	004D1 004D5		MOVZBL	1(LAST_OPERAND), -(SP) #2, DBG\$GET_BIF_ARGUMENTS R0, ARG_LIST aARG_LIST, #2 54\$	1
		6E 02	00	BE	01	004DA 004DD 004E1		CMPL	aARG_LIST, #2	7516
		01	00	BE	14 01 12	004E3		CALLS MOVL CMPL BGTR CMPL BNEQ	WARU_LISI, WI	7525
		7E		68	9A 11	004E7		MUVZBL	52\$ (TOKEN), -(SP)	7529
				02	00	004EC	52\$: 53\$:	BRB PUSHL	538 #3	7529 7528 7532
		7E	08 04	A6	9F	004F0 004F3	53\$:	PUSHAB	8(LAST_OPERAND) 4(LAST_OPERAND), -(SP)	
	0000v	7E CF 58		03	9F 9A FB DO	004EE 004F0 004F3 004F7 004FC		MOVZBL CALLS MOVL	4(LAST_OPERAND), -(SP) #3, CREATE_OPERATOR_TOKEN R0, TOKEN #8, ARG_LIST, R2	
52		58 6E		08	C1	004FF 00503		ADDL3 PUSHL	(86)	7535
53	04	AE		04	DD C1	00503 00505 0050A		ADDL3	W4, ARG_LIST, R3 (R3)	
	000000006	00		58	DD	00500		PUSHL	TOKEN AS DRESEVAL LANG OPERATOR	
	00000000	00	08	O2DF A6	DD	0050E 00515 00518	548:	BRW	#3, DBG\$EVAL_LANG_OPERATOR 102\$ 8(LAST_OPERAND)	7546
			00028838	01 8F	DD	0051B	,40.	MOVL ADDL3 PUSHL ADDL3 PUSHL PUSHL CALLS BRW PUSHAB PUSHL PUSHL	#1 #165944	
		01	02	00A7	DD DD 31 B1	0051B 0051D 00523 00526	55¢.	BRW	65\$	7563
		U	02	03	13	0052A	55\$:	BEQL	2(LAST_OPERAND), #1 56\$: 1703
			FF32	0092 C28 AE AE AE AE	13 31 95 12 9F 9F	0052A 0052C 0052F 00533 00538 0053B 0053E	56\$:	BRW CMPW BEQL BRW TSTB BNEQ PUSHAB	64\$ PATHDESC+2	7571
			20	AE	9F	00535		PUSHAB	58\$ DUMMY	7578
			20 24 20 30 08	AE	9F	00538 0053B		PUSHAB	DUMMY PRIMPTR	1
			3C 08		9F 9F	0053E 00541		PUSHAB PUSHAB PUSHAB PUSHAB	R(IAST OPERAND)	
	0000000G	00 0F		05 50	FB E9 D1 13	00544 0054B		BLBC	#5, DBG\$DEF_SYM_FIND RO, 58\$ KIND, #1 57\$	1
		01	30	AE 04	D1	0054E 00552		CMPL BEQL	KIND, #1	7583
		05	30	AE 03	D1	00554	57\$:	CMPL	KIND, #5	7584
				00AA	31	0055A	585:	BRW	58\$ 67\$ -(SP)	7602
			FF30	56	DD	0055F	58\$: 59\$:	PUSHL	LAST OPERAND PATHDESC	
	0000v	CF	1130	03	FB	00565		CALLS	#3, APPEND_TO_PATHNAME	7608
				7E	70	00560		CLRO	-(SP) -(SP)	1.000
			40	AE	9F	00570		PUSHAB	KIND	
	00000000	00	40 48 FF30	CD	9F	00576		PUSHAB	PATHDESC CETSYMBOL	
	0000000G	00	34	050 A04 00 00 00 00 00 00 00 00 00 00 00 00 0	D121314D9FBD7C4FFB5313F	00544BE240055554BE240055554BE240055554BE240055554BE240055554BE2400555554BE2400555554BE2400555554BE2400555554BE2400555554BE2400555554BE2400555554BE2400555554BE24005555554BE24005555554BE240055555555555555555555555555555555555		CALLS BLBC CMPL BEQL CMPL BNEQ BRW CLRL PUSHAB CALLS PUSHAB CALLS PUSHAB PUSHAB PUSHAB PUSHAB PUSHAB PUSHAB PUSHAB	KIND TYPEID PATHDESC #7, DBG\$STA_GETSYMBOL TYPEID	7613
				0262	31	00586	400	BRW	101\$	7414
			38	AE	91	00289	60\$:	LO2HAR	PATHSTRING	: 7616

					1	1 5-Sep-198 4-Sep-198	84 02:10 84 12:17	:13	VAX-11 Bliss-32 V4.0- [DEBUG.SRC]DBGPARSER.	742 B32;1	Page 256 (25)
0000000G	00	FF30	CD	9F FB	0058C 00590		PUSHAB CALLS CMPL BNEQ PUSHL PUSHL PUSHL	PATHO	DESC DBG\$NPATHDESC_TO_CS		:
00000000	09	30	AE OD AE O1 8F	D1 12	00597		CMPL	KIND.	, #9		7617
		38	AE	DD	0059B 0059D		PUSHL	PATHS	STRING		7619
		000281F0	8F	DD DD 11	005A0 005A2		PUSHL	#1643	336		
	OD	30	AE 03	D1	005A8 005AA	615:	BRB CMPL BEQL BRW PUSHL PUSHL PUSHL BRW MOVZBL PUSHL	63\$ KIND	, #13		7621
			0226	13	005AF		BEQL	62\$			
		38	AE 01	00	005B0 005B3 005B6	62\$:	PUSHL	PATHS	STRING		7623
		000282A8	8F	DD 31	005B8 005BE		PUSHL	#1645	520		1
	7E	08	0223 A6	9A	005C1	63\$: 64\$:	MOVZBL	100\$ 8(LAS	ST_OPERAND), -(SP)		7625
		000289E2	01 8F	DD	005C5 005C7		PUSHL	#1663			
0000000G	00		8F 03 43	FB	005CD 005D4	65\$:	CALLS	#3, L	LIB\$SIGNAL		7554
	01	02	A6	81	00506	66\$:	CMPW	2(LAS	ST_OPERAND), #1		; 7556 ; 7658
		FF32	A6 50 CD 3A	95	005DA 005DC		PUSHL CALLS BRB CMPW BNEQ TSTB	PATHO	DESC+2		: 7666
		20	3A AE	9F	005E0 005E2 005E5		BNEW	69\$ DUMMY	,		7673
		24	AE	9F	005E5 005E8		PUSHAB	DUMMY			
		20 24 20 30 08	AE	9F 9F	005EB 005EE 005F1		PUSHAB	KIND			
0000000G	00	00	05	FB E9	005F1		CALLS	#5. 0	ST_OPERAND) DBG\$DEF_SYM_FIND		
	00 21 01	30	AE	D1	005FB		CMPL	KIND,	, #1		7678
	05	30	AEE AGS OF AG OF AG	13	005FF 00601		PUSHAB PUSHAB PUSHAB PUSHAB CALLS BLBC CMPL BEQL CMPL	KIND.	. #5		7679
			10	D1 12 04	00601 00605 00607	675:	BNEQ	69\$ -(SP)			7688
		24	7E AE AE O4	9F 9F DD FB 31	00607 00609 0060C 0060F 00612	0.0.	PUSHAB	DUMMY			
		24 20 30	AE	DD	0060F		PUSHAB	PRIMP	PTR		
0000000G	00		02AD	FB 31	00612	685:	BRW	1225	DBG\$NCOPY_DESC		7681 7697
			02AD 7E 56 CD 03	D4	0061C 0061E 00620 00624 00629 0062C 00633 00634	68\$: 69\$: 70\$:	PUSHL CALLS BRW CLRL PUSHL PUSHAB CALLS BRW PUSHL	-(SP)	OPERAND		7697
0000v	CF	FF30	CD	DD 9F	00620		PUSHAB	PATHE	OPERAND DESC ARREND TO PATHNAME		
00004	Cr		0272	FB 31 DD FB 31 E9	00629	710	BRW	1155	APPEND_TO_PATHNAME		7698 7707
0000v	CF		56	f B	0062E	71\$:	CALLS	#1. T	OPERAND CONSTANT_TO_VALDESCR		: ""
	12	08	027A	31 E9	00633 00636	728:	BRW	116\$ NUMER	RIC_PATHNAME, 73\$: 7724
		80 80	AE A6	9F	0063A		PUSHAB	8(LAS	ST_OPERAND)		7724
00000000	00	0002898A	01 8F 03	DD DD FB 31	0063F		PUSHL	#1662	282		
0000000G	00		OOEE		0063A 0063D 0063F 00645 0064C	73\$: 74\$:	BRW	84\$	LIB\$SIGNAL		7728 7738
			7E 56	00	0064F 00651	748:	BRW BLBC PUSHAB PUSHL PUSHL CALLS BRW CLRL PUSHL	-(SP)	OPERAND		: 7758

					1	5-Sep-1 4-Sep-1	984 02:10 984 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 257 (25)
00004		FF30	CD	9F	00653		PUSHAB	PATHDESC	
0000v	CF	FE60	03 CD 57	FB 9F 9F FB	00657 0065C		PUSHAB	#3, APPEND TO PATHNAME SAVED PATHDEST	7739
			57 AF	DD	00660		PUSHL PUSHAB PUSHAB	SAVED PATHDEST PLIPTR SUBSCR_DESC	7740
00000		FF30	AE CD 04	9F	00662 00665 00669		PUSHAB	PATHDESC	: 1134
0000v	CF AE		50	DO	00669 0066E		MOVE	#4, PATHNAME_TO_PRIMARY RO, PRIMPTR	
			59	DO 11	0066E 00672 00674	75e.	MOVL BRB	79\$	7064
			7E 56 CD 03	00	00676		CLRL PUSHL	-(SP) LAST_OPERAND	1/01
0000v	CF	FF30	CD	PF FB DD PF	00678 00670		PUSHAB	PATHDESC #3, APPEND_TO_PATHNAME	
00001			01	DD	00681		PUSHL		: 7762
		FE60	CD 57	16	00683 00687		PUSHAB	SAVED_PATHDESC PLIPTR	7763
		FF30	AE	DD 9F	00689		PUSHAB	SUBSCR_DESC	7763
0000v	CF	7730	AE CD 05 50	9F FB DO	00689 00680 00690 00695		PUSHAB	PATHDESC #5, PATHNAME_TO_PRIMARY	
24	AE	24	50 AE	DO	00695	76\$:	MOVL PUSHL	RO, PRIMPTR PRIMPTR	7764
			0219	DD 31	0069C		BRW	118\$	
			7E	00	0069F 006A1	77\$:	CLRL PUSHL	-(SP) LAST_OPERAND	7787
0000v	CF	FF30	CD 03	DD 9F FB	006A3		PUSHAB	PATHDESC	
00000		FE60	CD 57	9F	006AC		CALLS PUSHAB	#3, APPEND TO PATHNAME SAVED PATHDEST PLIPTE	7788
		48	AE	DD 9F	006B0 006B2		PUSHL PUSHAB	SUBSCR_DESC	7789
0000v	CF	FF30	AE CD 04	9F FB	006B5 006B9		PUSHAB	PATHDESC	
24	AE		50	DO	006BE 006C2		MOVL	#4. PATHNAME_TO_PRIMARY RO, PRIMPTR	
		08 28	50 A6 AE 02	9F DD	006C2	78\$:	PUSHAB PUSHL	8(LAST_OPERAND) PRIMPTR	7790
0000v	CF			FB	00668	700	CALLS	#2, GET_BLISS_SUBSCRIPTS	
			7C 7E	04	006CD 006CF	79\$: 80\$:	BRB	86\$ -(SP)	7064
		FF30	56		006D1 006D3		CLRL PUSHL PUSHAB	LAST OPERAND	
0000V	CF		03	FB	00607		CALLS	#3. APPEND_TO_PATHNAME	
		FE60	57	9F DD	006DC 006E0		PUSHAB	SAVED PATHDESC PLIPTR	: 7804 : 7805
		FF30	7E 5CD3 CD7 AED 050 AE	DD 9 FB 9 FB DO 9 FB DO	006E2 006E5 006E9 006EE 006F2		PUSHL PUSHAB PUSHAB CALLS MOVL PUSHL	LAST OPERAND PATHDESC #3, APPEND TO PATHNAME SAVED PATHDESC PLIPTR SUBSCR DESC PATHDESC #4 PATHNAME TO PRIMARY	7804
0000v	CF	1130	04	FB	006E9		CALLS	WYA FAIRINGE TO FRIEND	
24	AE	24	50 AF	00	006EE	81\$:	MOVL	RO, PRIMPTR PRIMPTR 120\$	7806
			0109	31	006F5	010.	BRW	120\$	
			56	00	006FA	829:	BRW CLRL PUSHL PUSHAB	-(SP)	7822
0000v	CF	FF30	CD	9F	006FC 00700		PUSHAB	PATHDESC	
00004	CI	FE60	CD	031400FBF909F	00705		PUSHAB	LAST OPERAND PATHDESC #3. APPEND TO PATHNAME SAVED PATHDESC	: 7823
		48	AF.	DD 9F	00709 0070B		PUSHL PUSHAB	PLIPTR SUBSCR_DESC	7823 7824 7823
00004	ce	FF30	7E 50 03 057 AE 04	9F	00705 00709 0070B 0070E 00712		PUSHAB	PATHDESC	
0000v	CF		04	FB	00/12		CALLS	#4. PATHNAME_TO_PRIMARY	•

					1	1 -Sep-1 -Sep-1	984 02:10 984 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 EDEBUG.SRCJDBGPARSER.B32;1	Page 258 (25)
	57 06	07	50 A7	D0 91	00717 0071A		MOVL CMPB	RO, PLIPTR 7(PLIPTR), #6	7825
	10	06	50 A7 09 A7 03 014C	91 12	0071E 00720 00724		MOVL CMPB BNEQ CMPB BNEQ BRW	6(PLIPTR), #16	7826
		FF30	O14C AE CD	12 91 12 31 9F	00724 00726 00729 00720	83\$:	DIICMAM	83\$ 112\$ NAME PATHDESC	7831
000000006	00	28	0129 7E 12	FB DD 31	0072C 00730 00737 0073A		PUSHAB CALLS PUSHL BRW CLRL BRB PUSHAB	#2, DBG\$NPATHDESC_TO_CS NAME 111\$	7832
			7E	11	0073A 0073D 0073F	84\$:	CLRL BRB	-(SP) 89\$	7908
0000v	CF	08	A6 5A 02 11	9F DD FR	00741 00744 00746	85\$:	PUSHAB	8(LAST_OPERAND) R10 #2, GET_RECORD_COMPONENT	7917
			01	FB 11 DD 11	00746 0074B 0074D 0074F	86\$: 87\$:	PUSHL CALLS BRB PUSHL	#2, GET_RECORD_COMPONENT 90\$ #1 89\$	7926
		5570	02 02 56 CD 03 F 908	DD DD 9F	00751	88\$: 89\$:	PUSHL PUSHL PUSHAB CALLS BRW PUSHAB PUSHL CALLS BRW PUSHL	#2 LAST_OPERAND PATHDESC	7935
0000v	CF	FF30	F908	FB 31 9F	00753 00755 00759 0075E 00761 00764 0076B 0076B	90\$: 91\$:	CALLS	#5, APPEND_TO_PATHNAME	7064 7947
0000v	CF	08	A6 5A 02 0148	9F DD FB 31	00761 00764 00766	91\$:	PUSHAB PUSHL CALLS	8(LAST_OPERAND) R10 #2. GET RECORD COMPONENT	7947
			0148	31 DD 11	0076B 0076E	92\$:	BRW PUSHL	#2 GET_RECORD_COMPONENT	7948 7958
			01 02 7E 56 CD	04 00 9F	00772	93\$: 94\$:	CLRL	94\$ -(SP) LAST_OPERAND	7968
0000v	CF	FF30	03 009A	FB	00776 0077A 0077F		PUSHAB CALLS BRW	PATHDESC #3. APPEND_TO_PATHNAME 107\$	7969
00004	"	08	A6	31 9F DD	00782	95\$:	PIICHAR	8(LAST_OPERAND) R10	7969 7982
0000v	CF		0130 01 56	FB 31 DD	0077F 00782 00785 00787 0078C 0078F 00791 00793	96\$:	PUSHL CALLS BRW PUSHL PUSHL PUSHAB	#2, GET_RECORD_COMPONENT 119\$ #1	7983 7997
0000v	CF	FF30	56 CD 03	DD DD 9F	00791 00793 00797		PUSHL PUSHAB CALLS	LAST OPERAND PATHDESC #3. APPEND TO PATHNAME	
		FE60	57	FB 9F DD 9F	007A0		CALLS PUSHAB PUSHL	#3. APPEND TO PATHNAME SAVED PATHDESC PLIPTR SUBSCR_DESC	7998 7999
0000v	CF 57	FF30	CD 04	9F FB	007A2 007A5 007A9		PUSHAB PUSHAB CALLS	#4. PATHNAME_TO_PRIMARY	7998
	57 06	07	CD CD CD ST AE CD O4 SA7 O9 A7	91 12	007AE 007B1 007B5		MOVL CMPB BNFO	RO, PLIPTR 7(PLIPTR), #6 97\$	8000
	10	06	03	12 91 12	007BB 007BB		CALLS MOVL CMPB BNEQ CMPB BNEQ	6(PLIPTR), #16 97\$ 112\$	8001
000000006	00	FF30	00B5 AE CD 02	12 31 9F 9F FB	0078D 007C0 007C3 007C7	97\$:	BRW PUSHAB PUSHAB CALLS	NAME PATHDESC #2, DBG\$NPATHDESC_TO_CS	8006

0000

0270 8F

					1	1 6-Sep-19 4-Sep-19	84 02:10 84 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 259 (25)
		20	0092	DD 31	007CE		PUSHL	NAME 111\$: 8007
			01	DD	007CE 007D1 007D4	98\$:	PUSHL	#1	: 8027
		38	FD86	31 DD DD	007D6 007D9	99\$:	BRW PUSHL	59\$ PATHSTRING	8050
		000281F8	01	DD	007DC 007DE		PUSHL	#1 #164344	
0000000	00 00		AE 08538 558 050	FB	007E4	100\$:	PUSHL CALLS PUSHL	#3. LIB\$SIGNAL	9057
		38	AE	DD DD FB	007ED 007F0	1013:	PUSHL	TOKEN TYPE ID	8053
0000000	00G 00		50	FB DO	007F7	102\$:	PUSHL CALLS MOVL MOVL	#2, DBG\$EVAL_ADA_TICK RO. PRIMPTR	
1	8 BC		01	DO 11	007FB		MOVL BRB	#1 aRET_OPERAND_FLAG	8061
		08	0A A6 5A 02	9F	007FF 00801 00804	103\$:	PUSHAB	8(LAST_OPERAND)	8026 8074
000	OV CF		02	9F DD FB	00806 0080B		PUSHL	#2 GET_RECORD_COMPONENT	
			00BB	31 DD 31	0080B 0080E	104 \$:	BRW PUSHL	이 🕷 🔭 보는 사람들이 나타보다 그 아이들이 모든 사람이 되었다. 그리고 사람들이 살아보다 그리고 있다.	: 8073 : 8084
			FEOB 5A	31 DD	0080E 00810 00813		BRW PUSHL	70\$ R10	8073 8084 8096 8109
000	OOV CF		01 77	FB	00815	1000.	CALLS	#1 FIX_UP_PRIMARY	10107
		40	AE	9F	00813 00815 0081A 0081C	107\$:	BRB PUSHAB PUSHAB	SUBSCR_DESC PATHDESC	8136
000	OV CF	FF30	02	9F FB	0081F 00823 00828		CALLS	M2. SAVE_SUBSCRIPTS	
			AE CD 02 69 5A	11 00	85800 A5800	108\$:	CALLS BRB PUSHL	#2, SAVE_SUBSCRIPTS 1138 R10	: 7064 : 8149
000	OV CF		01	DD FB 31 9F	0082C 00831	1000.	CALLS	#1 FIX_UP_PRIMARY	
		FE60	008B CD 57	9F	00834	109\$:	BRW PUSHAB	SAVED PATHDESC PLIPTR	: 8150 : 8164
		48	AE	DD 9F	00838 0083A		PUSHL PUSHAB	SUBSCR_DESC	8164 8165 8164
000	OV CF	FF30	CD 04	9F	0083D 00841		PUSHAB	PATHDESC	
000	57 06	0.7	50 A7	DO	00846		MOVL	#4, PATHNAME_TO_PRIMARY RO, PLIPTR 7(PLIPTR), #6	9144
			06	91	UUSCD		BNEQ	110\$ 6(PLIPTR), #16	8166
	10	06	A7 20	91	0084F 00853 00855 00858 00863 00866		MOVL CMPB BNEQ CMPB BEQL PUSHAB	6(PLIPTR), #16 112\$	8167
		FF30	AE	9F 9F	00855	110\$:	PUSHAB PUSHAB	112\$ NAME PATHDESC	8172
0000000	00 00		02		9085C		CALLS	#2, DBG\$NPATHDESC_TO_CS	9177
		30	01	FB DD DD DD FB 04	00866	1115:	PUSHL	NAME #1	8173
0000000	00G 00	00028CA0	8F 03	DD	00868 0086E 00875		PUSHL	#167072 #3, LIB\$SIGNAL	
		24 08	AE	04	00875	112\$:	CLRL	PRIMPTR NUMERIC PATHNAME	: 8178 : 8179
00	6E		00	D4 20	0087B		CLRL MOVC5	PRIMPTR NUMERIC PATHNAME NO, (SP), NO, N208, PATHDESC	8180
	5B 6E	FF30 FF38	CD	9E	00885		MOVAB	PATHDESC+8, PATHVECTOR #0, (SP), #0, #624, SUBSCR_DESC	8181 8182
00	6E	40	00	50	00878 00878 00882 00885 0088A 00891 00893		MOVC5		:
			31	11	00893	113\$: 114\$:	BRB PUSHL	121\$ R10	7064
000	OOV CF		ÓÎ	FB	00895 00897		CALLS	#1, FIX_UP_PRIMARY	: "

DBGPARSER V04-000				M 1 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 260 (25)
			FE60 CD 57 48 AE FF30 CD 04	E 9F 008A4 PUSHAB SUBSCR DESC	: 8193 : 8204 : 8205 : 8204
	0000v	CF AE	04 50 13	4 FB 008AB CALLS #4, PATHNAME_TO_PRIMARY 0 D0 008B0 116\$: MOVL RO PRIMPTR 3 11 008B4 BRB 122\$	8203 8224
	0000v		01 07 5A 01	7 11 009BB 1185: CALLS #1, GET_SUBSCRIPTS 7 11 009BD BRB 121\$ A DD 008BF 1198: PUSHL R10	8224
	0000V 14	CF BC	01 F7A0 24 AE	1 FB 008C1 1205: CALLS #1, GET_DEREFERENCE 0 31 008C6 1215: BRW 45 E DO 008C9 1225: MOVL PRIMPTR, DRET_TOKEN 04 008CE RET	8259 8262

; Routine Size: 2255 bytes, Routine Base: DBG\$CODE + 1988

```
8310
8311
8312
8313
8314
8316
8317
8318
```

This routine appends a pathname component to an existing Pathname Descriptor. It is called by the Primary Parser during the parsing of the pathname part of Primary Symbols (e.g., A\B\C) to build up the Pathname Descriptor which must eventually be passed to GETSYMBOL to get the symbol's SYMID.

ROUTINE APPEND_TO_PATHNAME(PATHDESCR, TOKEN, COMPONENT_KIND): NOVALUE =

INPUTS

PATHDESCR - A pointer to a Pathname Descriptor to which a new pathname component should be appended.

TOKEN - A pointer to the Lexical Token Entry for the identifier to be appended to the pathname descriptor.

COMPONENT_KIND - This tells us what kind of pathname component we are dealing with. It can have one of the following values:

NOT_REC_COMP (0) - An ordinary pathname component to be appended onto the end of the pathname, e.g., "A" and "B" would be of this kind

REC_COMP (1) - An ordinary pathname component to be appended onto the end of the pathname, e.g., 'A' and 'B' would be of this kind in the pathname 'A\B.C'

REC_COMP (1) - In the above example, 'C' comes in as a record component. It is appended to the pathname and the TOTCNT is incremented but not the PTHCNT field.

COB_REC_COMP (2) - In COBOL, record components come in first, e.g., "C of B of A", so we have to treat them differently when we are building the pathname.

OUTPUTS

The identifier specified by TOKEN is added to the end of the Pathname Descriptor pointed to by PATHDESCR.

BEGIN

MAP

PATHDESCR: REF PTH\$PATHNAME, TOKEN: REF TOKEN\$ENTRY; ! Pointer to Pathname Descriptor ! Pointer to Identifier Token Entry

LOCAL

PATHVECTOR: REF VECTOR[,LONG]; ! Pointer to pathname vector in the PATHDESCR Pathname Descriptor

Make sure we have a valid Identifier Lexical Token Entry. Also make sure it will fit in the Pathname Descriptor.

IF .TOKEN[TOKEN\$W_CODE] NEQ TOKEN\$K_IDENTIFIER

SIGNAL (DBG\$_ILLPATHELEM, 1, TOKEN[TOKEN\$B_LENGTH]);

IF .PATHDESCR[PTH\$B_PATHCNT] GEQ DBG\$K_MAX_PATHNAME THEN

We want to insert the name at the PATHCNT position. Push down lower names (record component names).

V0	GPARS 4-000 8269 8270 8271 8273 8274 8275 8276 8277 8278	ER		837 837 838 838 838 838 838 838 838 838	78901234567	55555555	ENC	PAT PAT PAT RET END	HVECHDES	t th	ne gi	ven	name	, in	B_TOTCI ECTORE cremen	NT] TO .I-1];	ounts, an	### PAX-11 Bliss-32 V4.0-742 ### PATHCNT]+1 DO ### REPTH\$B_PATHCNT]+1 DO #### REPTH\$B_LENGTH]; ####################################	Page 263 (26)
																	.PSECT	DBG\$PLIT,NOWRT, SHR, PIC,0	
45	45	40	41	4E	48	54	41	52	5F	50 4F	54	5F 30	44	1F 4E 20	031DB 031EA	P.AXL:	.ASCII	<31>\DBGPARSER\<92>\APPEND_TO_PATHNAME\	
									62	54 66 53 66 50 65 50 50 60 60 60 60 60 60 60 60 60 60 60 60 60	00000	08 02 08 98A 04 01 08 9D2 08 0C 08	00C4E41F3C3CA0A0A0A49CC31F1F1F3	907 PEO DE BON D	00002 00009 00001 000113 000118 000018 000029 000029 000030 000039 000040 00040 00040 00053 00058		PSECT TO PATHN MOVAB MOVAB MOVAB MOVAB PUSHAB PUSHAB PUSHAB PUSHAB PUSHAB PUSHAB CMPB BASSHAB PUSHAB CMPB BNEQ BNEQ BNEQ BNEQ BNEQ BNEQ BNEQ BNE	DBG\$CODE,NOWRT, SHR, PIC,O IAME: Save R2,R3,R4,R5,R6 LIB\$SIGNAL, R6 TOKEN, R4 2(R4), #1 1\$ 8(R4) #1 #166282 #3, LIB\$SIGNAL PATHDESCR, R3 1(R3), #50 2\$ 8(R4) #1 #166354 #3, LIB\$SIGNAL 8(R3), PATHVECTOR COMPONENT_KIND, #1 3\$ (R3), R0 8(R4), (PATHVECTOR)[R0] 10\$ COMPONENT_KIND, #2 7\$ 1(R3) 4\$ P.AXL #1 #164706 #34, LIB\$SIGNAL	8263 8314 8316 8318 8320 8322 8326 8333 8334 8341 8347 8349

6240 FC A F7 62 08	06 11 0006E A240 D0 00070 5\$: MOVL -4(PATHVECTOR)[I], (PATHVECTOR)[I] 50 F5 00076 6\$: SOBGTR I, 5\$ A4 9E 00079 MOVAB 8(R4), (PATHVECTOR)	0750
0C 51 01 55 01 6240 FC A 55 6241 08 01	27 11 0007D BRB 10\$ AC D5 0007F 7\$: TSTL COMPONENT_KIND 24 12 00082 BNF0 11\$	8358 8362 8370 8377 8378 8383 8384 8384 8387

; Routine Size: 169 bytes, Routine Base: DBG\$CODE + 225D

```
ROUTINE CHECK_UPSCOPE(SYMID, TYPEID, PRIMPTR, DEPTH) =
                                        FUNCTION
                                                 This routine is used to implement "incomplete data qualification" in language BASIC. For example, a user can abbreviate A::B::C with A::C. We are passed in a SYMID for the record component ("C" in the above example), and a TYPEID for the record ("A" in the above example. This routine determines whether we can get to the record by going upscope from the component. If so, it returns TRUE and modifies the Primary to include the intervening component
                                                  selection.
                                        INPUTS
                                                  SYMID - SYMID for the record component.

TYPEID - TYPEID for the record.
                                                  PRIMPTR - Pointer to the Primary being constructed.
                                                  DEPTH - recursion depth
                                        OUTPUTS
                                                  The value TRUE is returned if the component selection is valid.
                                                  In this case, the Primary is modified to include the intervening
                                                  component selection.
                                                  The value FALSE is returned if the component is not a valid
                                                  component for this record.
                                  BEGIN
                                            MAP
                                                  SYMID: REF RSTSENTRY,
TYPEID: REF RSTSENTRY
                                                  PRIMPTR: REF DBG$PRIMARY;
                                           LOCAL
                                                  COMPSYMID,
                                                 NODEPTR: REF DBGSPRIM NODE,
                                                  TYPCOMPLST: REF VECTOR[,LONG],
TYPREFTBL: REF VECTOR[,LONG],
                                                  U_SYMID: REF RSTSENTRY;
                                            DBG$GL_CURRENT_PRIMARY = .PRIMPTR;
                                              Check that the RST entry upscope from the given record component SYMID is a Type RST Entry.
Then check whether it matches the given TYPEID for the record.
                                           U_SYMID = .SYMID[RST$L_UPSCOPEPTR];
IF .U_SYMID[RST$B_KIND] NEQ RST$K_TYPE
THEN
                                           RETURN FALSE:
IF .U_SYMID EQL .TYPEID
THEN
                                                  BEGIN
                                                     The TYPEID matches, so the record component SYMID is valid.
                                                   ! In this case, we do want to perform the component selection.
```

```
TYPREFTBL = .U_SYMID[RST$L_TYPREFTBL];
I = 0:
WHILE .TYPREFTBLE.IJ NEQ 0 DO
    BEGIN
    U_SYMID = .TYPREFTBL[.1];
IF CHECK_UPSCOPE(.U_SYMID, .TYPEID, .PRIMPTR, .DEPTH+1)
     THEN
         BEGIN
```

! If we are at the top level of recursion, then do not modify ! the Primary here. This is because it will get done in

Page 266 (27)

```
GET_RECORD_COMPONENT.
                                          .DEPTH EQL O
                                       THEN
                                           RETURN TRUE:
                                         The TYPEID matches, so the record component SYMID is valid.
                                         In this case, we do want to perform the component selection. Modify the Primary to include the record component selection.
                                      NODEPTR = .PRIMPTR[DBG$L PRIM_BLINK];
NODEPTR[DBG$V PNODE EVAL] = TRUE;
TYPCOMPLST = TYPEID[RST$A_TYPCOMPLST];
                                         Determine the "component index". This "PNREC_INDEX" field gets
                                         used in determining the logical successor.
                                       INCR I FROM O TO .TYPEID[RST$L_TYPCOMPCNT] - 1 DO
                                           COMPSYMID = .TYPCOMPLST[.1];
                                              If this component is the one we seek, set its index into the Record
                                              Sub-Node and leave the loop.
                                            IF .SYMID EQL .COMPSYMID
                                            THEN
                                                BEGIN
                                                NODEPTREDBG$W_PNREC_INDEX] = .I + 1;
                                                EXITLOOP:
                                                END:
                                           END:
                                         Check for FCODE of record - otherwise, it is not valid to skip
                                         this component.
                                       DBG$STA_SYMTYPE(.SYMID, FCODE, NEW_TYPEID);
                                       IF .FCODE NEQ RSTSK_TYPE_RECORD
                                           RETURN FALSE:
                                         finally append another Primary Descriptor Sub-Node for the selected
                                         record component. Then return.
                                       DBG$BUILD_PRIMARY_SUBNODE (.PRIMPTR, RST$K_TYPCOMP, .SYMID,
                                                                     .FCODE, .NEW_TYPEID, O);
                                       RETURN TRUE:
                                       END:
                                  I = .I + 1;
                                  END:
                                We failed to find a path to the desired TYPEID. Return FALSE.
                              RETURN FALSE:
                              END:
```

			OFFC	00000	CHECK_U	PSCOPE:			
		5E				WORD SUBL2 MOVL MOVL MOVL CMPB	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 #8, SP	: 8	3388
	000000006	5E 56	0C AC DO 56 DO 10 AB DO 14 A7 91	00002 00005 00009		MOVL	#8, SP PRIMPTR, R6 PG DRGSGL CURRENT PRIMARY	. 8	3429
	00000000	58	04 AC DO	00010		MOVL	R6, DBG\$GL_CURRENT_PRIMARY SYMID, R8 16(R8), U_SYMID	: 8	3435
		07	0C AC DO 56 DO 10 A8 DO 14 A7 91			CMPB	20(U_SYMID), #7	: 8	3436
			03 13 00A6 31	0001C		BRW	20(U_SYMID), #7 1\$ 12\$	•	
		55	08 AC DO	00021	1\$:	MOVL CMPL BNEQ	TYPEID, RS U_SYMID, RS	. 8	3439
			21 12	00028		BNEQ	45	1.	
	0A	A3	18 A6 D0 01 88 20 A5 9E	0002A 0002E 00032		BISB2	24(R6), NODEPTR #1, 10(NODEPTR)	: 8	3447
		53 63 52 50	18 A6 D0 01 88 20 A5 9E 01 CE 09 11	00032		MOVL BISB2 MOVAB MNEGL	44(R5), TYPCOMPLST	. 8	3449
			09 11 6240 D0	00036 00039 0003B	2\$:	BRB MOVL	#1, I 3\$ (TYPCOMPLST)[I], COMPSYMID		
		59 59	58 01	0003F	20.	CMPL	R8, COMPSYMID	: 8	8462
	F2	50	58 D1 48 13 28 A5 F2 4D 11 1C A7 D0	00042	3\$:	BEQL AOBLSS	7\$ 40(R5), 1, 2\$		3454
		5A	1C A7 D0	00049 0004B 0004F	45:	BRB	28(U_SYMID), TYPREFTBL	: 8	3454 3473 3491 3492 3496 3493
	5B 10	AC	54 D4 01 C1	0004F 00051		CLRL		: 8	1492
			6A44 D5	00056	5\$:	TSTL	(TÝPREFTBL)[I]	. 8	1493
		57	6A44 D0	0005B		TSTL BEQL MOVL PUSHL MOVQ PUSHL CALLS	12\$ (TYPREFTBL)[I], U_SYMID	: 8	3495
		7E	5B DD 55 7D 57 DD	0005F 00061		MOVO	R5, -(SP)	: 0	1490
	96	AF	57 DD 04 FB	00064		PUSHL	WZ. CHECK_UPSCOPE		
		AF 56	50 E9	0006A		BLBC TSTL	RO, 118 DEPTH 108		3504
		67	10 AC DS			BEQL	10\$:	
	0A	53 A3	18 A6 D0 01 88			MOVL BISB2	24(R6), NODEPTR #1, 10(NODEPTR)	: 8	512
		52	2C A5 9E	0007A 0007E		MOVAB	#1, I	: 8	514 521
			10 11	00081	65:	BRB	8\$ (TYPCOMPLST)[I], COMPSYMID		
		59 59	6240 DO 58 D1 07 12	00087		BRB MOVL CMPL BNEQ ADDW3	R8. COMPSYMID	8	3527
18	A3	50	01 A1	00080	78:	ADDW3	8\$ #1, I, 24(NODEPTR)	: 8	530
	EB	50	28 A5 F2	00091	8\$:	AOBLSS	9\$ 40(R5), 1, 6\$: 8	530 529 519 538
			05 11 28 A5 F2 5E D0 08 AE 9F 58 D0 03 FE 04 AE D1	0007A 0007E 00081 00083 00087 0008C 00091 00098 0009A 0009B	8\$: 9\$:	BRB AOBLSS PUSHL PUSHAB PUSHL CALLS CMPL	FCODE	: 8	1538
	0000000G	00	58 DD 03 FB	00090		PUSHL	R8 #3, DBG\$STA_SYMTYPE FCODE, #7		
	00000000	00	04 AE D1	000A6		CMPL	FCODE, #7	: 8	3539

DBGPARSER V04-000			I 2 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 269 (27)
DC	A7 CF 50	04 0C	1B 12 000AA	8547 8548 8547 8549 8552 8493 8558

; Routine Size: 202 bytes, Routine Base: DBG\$CODE + 2306

```
8590
```

```
This routine converts a Constant Lexical Token Entry into the corresponding Value Descriptor. For string and character constants, this results in an ordinary Value Descriptor, but for numeric constants it results in an "unconverted" Value Descriptor, i.e., a descriptor in which the numeric constant is represented as the original input character string. The actual conversion of a numeric constant to its binary representation is thus delayed until the appropriate precision of the binary representation can be determined from context.
```

INPUTS

TOKEN - A pointer to a Constant Lexical Token Entry. This entry represents a numeric, string, or character constant of some sort.

OUTPUTS

A Value Descriptor is constructed for the constant and a pointer to that descriptor is returned as the routine value.

BEGIN

MAP

TOKEN: REF TOKENSENTRY;

ROUTINE CONSTANT_TO_VALDESCR(TOKEN) =

! Pointer to Lexical Token Entry

LOCAL

VALPTR: REF DBG\$VALDESC:

! Pointer to Value Descriptor we build

Build a skeleton Value Descriptor for the constant and copy the constant character representation into that descriptor.

```
VALPTR = DBG$MAKE_SKELETON_DESC(DBG$K_VALUE_DESC, .TOKEN[TOKEN$B_LENGTH]);
VALPTR[DBG$B_DHDR_LANG] = .DBG$GB_LANGUAGE;
VALPTR[DBG$B_DHDR_KIND] = RST$K_DĀTA;
VALPTR[DBG$B_DHDR_FCODE] = RST$K_TYPE_ATOMIC;
VALPTR[DBG$V_DHDR_UNCVT] = TRUE;
VALPTR[DBG$B_VALUE_CLASS] = DSC$K_CLASS_S;
VALPTR[DBG$W_VALUE_LENGTH] = .TOKEN[TOKEN$B_LENGTH];
VALPTR[DBG$W_VALUE_POINTER] = VALPTR[DBG$A_VALUE_ADDRESS];
VALPTR[DBG$W_VALUE_TOKENCODE] = .TOKEN[TOKEN$W_CODE];
CH$MOVE(.TOKEN[TOKEN$B_LENGTH], TOKEN[TOKEN$A_NAME],
VĀLPTR[DBG$A_VALUE_ADDRESS]);
```

Determine what kind of constant this is and set the data type and the 'unconverted' bit accordingly.

CASE .TOKEN[TOKEN\$W_CODE] FROM TOKEN\$K_MIN_OPERAND TO TOKEN\$K_MAX_OPERAND OF

Handle character string constants. Mark the data as type T.

```
[TOKEN$K_STRING]: VALPTR[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_T;
Handle bit string constants. Mark the data as type Tf.
                                  CTOKENSK_BIT_STRING]:
VALPTREDBGSB_VALUE_DTYPE] = DSCSK_DTYPE_V;
                                    Handle decimal integer constants. Mark the data as type L.
                                  LTOKENSK_INTEGER]:
                                       VALPTREDBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
                                    Handle hexadecimal integer constants. Mark the data as type L.
                                  CTOKENSK_HEX_INTEGER]:
VALPTREDBGSB_VALUE_DTYPE] = DSCSK_DTYPE_L;
                                    Handle octal integer constants. Mark the data as type L.
                                  LTOKEN$K_OCT_INTEGER]:
                                       VALPTREDBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
                                    Handle binary integer constants. Mark the data as type L.
                                  CTOKENSK_BIN_INTEGER]:
                                       VALPTREDBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
                                    Handle floating-point constants without exponents. Mark the data
                                    type as f.
                                  [TOKEN$K_FLOATING]:
                                       VALPTREDBG$B_VALUE_DTYPE] = DSC$K_DTYPE_F;
                                    Handle floating-point constants with E exponents. Mark the data
                                    type as f.
                                  CTOKENSK_EXP_E_FLOAT]:
VALPTR[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_F;
                                    Handle floating-point constants with D exponents. Mark the data
                                    type as D.
                                  TOKENSK EXP D FLOAT :

IF .DBG$GB_MOD_PTR[MODE_G_FLOATS]
                                           VALPTR[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_G;
VALPTR[DBG$W_VALUE_TOKENCODE] = TOKEN$K_EXP_G_FLOAT;
```

```
DBGPARSER
V04-000
                                                                                                                               VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1
                                                                                                                                                                                   Page 272
(28)
 8888888888888888888888888901234566001234566600123456660012345666001234566600123456660012345666001234566600123456660012345666600123456666001234566660012345666600123456666001234566666001234566666001234566666600123456666666
                                                          END
                                                   ELSE
                                                          VALPTR[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_D;
                                                 Handle floating-point constants with G exponents. Mark the data
                                                 type as G.
                                              CTOKENSK_EXP_G_FLOAT]:
VALPTRCDBGSB_VALUE_DTYPE] = DSCSK_DTYPE_G:
                                                 Handle floating-point constants with Q exponents. Mark the data
                                                 type as H.
                                              CTOKENSK_EXP_Q_FLOAT]:
VALPTRCDBG$B_VALUE_DTYPE] = DSC$K_DTYPE_H;
                                                Handle pack decimal constants. Mark the data type as P.
                                              TOKEN$K_PACK_DECIMAL]:
    VALPTR[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_P;
                                                Any other case is an internal error.
                                              [INRANGE, OUTRANGE]:
                                                   $DBG_ERROR('DBGPARSER\CONSTANT_TO_VALDESC');
                                              TES:
                                          The Value Descriptor is constructed. Return its address to the caller.
                                        RETURN . VALPTR:
                                        END;
                                                                                                                      DBG$PLIT, NOWRT, SHR, PIC, 0
                                                                                                           .PSECT
                                                                                      031FB P.AXM:
0320A
03218
          4F 43 5C 52 45 53
45 44 4C 41 56 5F
                                             52
4F
                                                                                                          .ASCII
                                                                                                                      <29>\DBGPARSER\<92>\CONSTANT_TO_VALDES\
                                                                                                           .ASCII \C\
                                                                                                           .PSECT DBG$CODE,NOWRT, SHR, PIC,0
                                                                              O1FC 00000 CONSTANT_TO_VALDESCR:
WORD Save R2,R3,R4,R5,R6,R7,R8
TOKEN, R8
TOKEN, R8
                                                                                                                                                                                        8559
8594
                                                        58
7E
7E
```

DBGPARSER V04-000	M 2 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 EDEBUG.SRCJDBGPARSER.B32;1	Page 273 (28)
0043 004F 0020 0020	00000000G 00	8595 8595 8596 8600 8600 8600 8600 8600 8600
	00000000	8617 8623 8647 8661 8668 8671 8672 8668 8676 8668
	02 A7 1C 90 000BF 10\$: MOVB #28, 2(R7) 04 11 000C3 BRB 12\$ 02 A7 15 90 000C5 11\$: MOVB #21, 2(R7) 50 56 00 000C9 12\$: MOVL VALPTR, R0 04 000CC RET	869 869 870 871

DBGPARSER V04-000

N 2 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1

Page 274 (28)

; Routine Size: 205 bytes, Routine Base: DBG\$CODE + 23D0

```
B 3
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                                   VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
                                                                                                                                                                                                              Page 275
(29)
   ROUTINE CONVERT_TO_INTEGER (VALPTR) =
                                           FUNCTION
                                                     This routine converts a value descriptor to an integer value
                                                     and returns the integer value.
                                           INPUTS
                                                     VALPTR -
                                                                                A pointer to a value descriptor
                                           OUTPUTS
                                                     An integer value is returned.
                                              BEGIN
                                              LOCAL
                                                     NEW_VALPTR: REF DBG$VALDESC:
                                                                                                           ! New value descriptor
                                                 Build a new value descriptor of type longword integer.
                                              NEW_VALPTR = DBG$MAKE_SKELETON_DESC(DBG$K_VALUE_DESC, 4);
NEW_VALPTR[DBG$B_DHDR_KIND] = R$T$K_DATA;
NEW_VALPTR[DBG$B_DHDR_FCODE] = R$T$K_TYPE_ATOMIC;
NEW_VALPTR[DBG$B_VALUE_CLASS] = DSC$K_CLASS_S;
NEW_VALPTR[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
NEW_VALPTR[DBG$B_VALUE_LENGTH] = 4;
NEW_VALPTR[DBG$W_VALUE_LENGTH] = 4;
NEW_VALPTR[DBG$L_VALUE_POINTER] = NEW_VALPTR[DBG$A_VALUE_ADDRESS];
                                               ! Call the routine which does type conversion.
                                              NEW_VALPTR = DBG$EVAL_LANG_OPERATOR (
                                                                  DBG$GL_CONVERT_TOKEN, .VALPTR, .NEW_VALPTR);
                                                 Return the value in the new value descriptor.
                                               RETURN .NEW_VALPTR [DBG$L_VALUE_VALUEO];
                                              END:
                                                                                           0000 00000 CONVERT_TO_INTEGER:
                                                                                                                                                                                                                     8712
8732
                                                                                                                                         Save nothing
                                                                                                   00002
                                                                                                                           PUSHL
                                                                                        04F2F800CF30
                                                                                               #122, -(SP)
#2, DBG$MAKE_SKELETON_DESC
#1538, 6(NEW_VALPTR)
#17301508, 20(NEW_VALPTR)
32(R0), 24(NEW_VALPTR)
NEW_VALPTR
VALPTR
VALPTR
VALPTR
                                                                                                                           MOVZBL
                                                                                7A
                                                                 00
A0
A0
                                               0000000G
                                                                                                    00008
                                                                                                                           CALLS
                                                                                                   0000F
00015
0001D
00022
00024
00027
00025
                                                                     01080004
                                                                                                                           MOVW
                                                                                                                           MOVL
                                                                                                                           MOVAB
                                                                                                                           PUSHL
```

PUSHL

CALLS

MOVL RET

PUSHAB

DBG\$GL_CONVERT_TOKEN #3. DBG\$EVAL_LANG_OPERATOR 32(NEW_VALPTR), RO

00000000

20

0000000G

D

8743

8748 8749

C 3 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1

Page 276 (29)

; Routine Size: 57 bytes, Routine Base: DBG\$CODE + 249D

007C 00000 CREATE_OPERAND_TOKEN: . WORD Save R2, R3, R4, R5, R6 MOVZBL aTOKENBUFFER, RO

: 8750 : 8792

RETURN . TOKEN;

END:

DBGPARSER V04-000							16-Si 14-Si	8 ep-1984 02:10 ep-1984 12:17):13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRCJDBGPARSER.B32;1	Page 278 (30)
	08	0 A6	0000000G 02 08	50 00 56 666 A66 50 BC	03 04 08	04 01 50 01 ACC 50 50 50	C6 00006 9F 00009 FB 0000C D0 00013 90 00016 B0 00019 9A 0001E D6 00022 28 00024 D0 0002A 04 0002D	DIVL2 PUSHAB CALLS MOVL MOVB MOVW MOVZBL INCL MOVC3 MOVL RET	#4, R0 3(RO) #1, DBG\$GET_TEMPMEM RO, TOKEN #1, (TOKEN) TOKEN_CODE, 2(TOKEN) aTOKENBUFFER, RO RO RO, aTOKENBUFFER, 8(TOKEN) TOKEN, RO	8793 8794 8795 8796 8798

; Routine Size: 46 bytes, Routine Base: DBG\$CODE + 24D6

TOKENBUFFER: REF VECTOR[, BYTE]; ! Pointer to token Counted ASCII string

! Pointer to new Token Entry created

VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32:1

Get a temporary memory block for the Operator Token Entry. Fill in the fields of the Token Entry and return its address.

TOKEN = DBG\$GET_TEMPMEM(TOKEN\$K_ENTSIZE_OPERATOR + .TOKENBUFFER[0]/%UPVAL + 1);
TOKEN[TOKEN\$B_KIND] = .KIND;
TOKEN[TOKEN\$W_CODE] = .TOKEN_CODE;
CH\$MOVE(.TOKENBUFFER[0] + 1, TOKENBUFFER[0], TOKEN[TOKEN\$B_OPLEN]);
RETURN .TOKEN;

END:

DBGPARSER V04-000		G 3 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 CDEBUG.SRCJDBGPARSER.B32;1	Page 280 (31)
	00000000G 00 56 66 02 A6 50 0C A6 08 BC 50	007C 00000 CREATE_OPERATOR_TOKEN: .WORD	8799 8844 8845 8846 8847 8848 8850
; Routine Size:	47 bytes, Routine Base:	DBG\$CODE + 2504	

```
H 3
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1
                          ROUTINE CREATE_PRID_CONSTANT(PRID) =
  FUNCTION
                                   This routine creates a value descriptor for the predefined identifier
                                   constant and returns the pointer to the value descriptor.
                            INPUTS
                                   PRID
                                            - Pointer to a Predefined Identifier Constant entry in Predefined Identifier Table for the given language.
                            OUTPUTS
                                   A pointer to the value descriptor of Predefined Identifier Constant is returned.
                               BEGIN
                                   PRID: REF PRIDSENTRY;
                                                                      ! Pointer to Predefined Identifier
                              VALPTR: REF DBG$VALDESC;
                                                                      ! Pointer to Value Descriptor
                              END:
```

			()00C	00000	CREATE_PRID_CON	STANT:		
			04	DD	00002	WORD PUSHL MOVZBL	41		8851 8875
000000006	7E	7A	8F 02	9A FB	00004	CALLS	#122, -(SP) #2, DBG\$MAKE_SKELETON_DESC		
	52		50	90	0000F	MOVL	RO, VALPTR		
03 07	AZ	0000000G	00		00012	MOVL MOVB MOVB	DBGSGB_LANGUAGE, 3(VALPTR)	:	8876
07	53	04	06 AC	90	0001A	MOAR	W6, 7(VALPTR) PRID, R3		8876 8877 8878
06	AZ	04 02 01	A3	90	00022	MOVE	PRID, R3 2(R3), 6(VALPTR)	;	0010
06 16	A2	ŎĪ	A3	90	00027	MOVB	1(R3), 22(VALPTR)		8879
	70	01	7Ę	04	00055	CLRL	-(SP)	:	8879 8880 8881
00000000	00	01	A3	FB	00035	MOVZBL	1(R3), -(SP) #2, DBG\$MAP_DTYPE_CLASS		8881
17			02	90	00035	CALLS	RO, 23(VALPTR)		
	A2 7E	01	A3	94	0003D	MOVZBL	1(R3), -(SP)		8882

Page 281 (32)

DBGPARSER V04-000					16-Se 14-Se	p-1984 02:10:1 p-1984 12:17:	13 VAX-11 Bliss-32 V4.0-742 30 CDEBUG.SRCJDBGPARSER.B32;1	Page 282 (32)
	00000000G 14 18 20	00 A2 A2 A2 50	20 04	01 50 A2 A3 52	FB 00041 B0 00048 9E 0004C D0 00051 D0 00056 04 00059	MOVAB MOVAB MOVL MOVL RET	#1, DBG\$NUM_BYTES RO, 20(VALPTR) 32(VALPTR), 24(VALPTR) 4(R3), 32(VALPTR) VALPTR, RO	8883 8884 8885 8886

; Routine Size: 90 bytes, Routine Base: DBG\$CODE + 2533

```
8887
8888
8889
8890
8891
8892
8893
   ROUTINE DUMP_OPERATOR(OPERATOR, PRIMARY_FLAG): NOVALUE =
FUNCTION
                                                 This routine dumps out an Operator Lexical Token Entry in a readable format when that operator is about to be evaluated. It prints one
                                                 line of output of approximately this format:
                         8894
8895
8896
8897
                                                                        primary operator "\" evaluated (infix)
                                                This routine is used only for internal DEBUG debugging purposes. Its output is not seen by normal DEBUG users. It is only called when Developer Switch 3 is set to trace operator evaluations as they occur.
                                       INPUTS
                                                OPERATOR - A pointer to the Lexical Token Entry for the operator to be dumped.
                                                PRIMARY_FLAG - A flag whose value is TRUE if this operator is evaluated as part of a Primary Symbol. Its value is FALSE if it is an ordinary expression operator.
                                       OUTPUTS
                                                NONE
                                          BEGIN
                                                OPERATOR: REF TOKENSENTRY:
                                                                                                ! Pointer to operator's token entry
                                             Print the main text of the message including the operator name string.
                                         DBG$PRINT(UPLIT BYTE(%ASCIC '), 0);
IF .PRIMARY_FLAG THEN DBG$PRINT(UPLIT BYTE(%ASCIC 'primary '), 0);
DBG$PRINT(UPLIT BYTE(%ASCIC 'operator '!AC' evaluated ('),
OPERATOR[TOKEN$B_OPLEN]);
                                             Then print what kind of operator this is.
                                           IF .OPERATOR[TOKEN$B_KIND] EQL TOKEN$K_PREFIX_OP
                                           THEN
                                                DBG$PRINT(UPLIT BYTE(%ASCIC 'prefix'), 0)
                                          ELSE IF .OPERATOR[TOKEN$B_KIND] EQL TOKEN$K_INFIX_OP
                        8934
8935
8936
8937
8938
8940
8941
8943
                                                DBG$PRINT(UPLIT BYTE(%ASCIC 'infix'), 0)
                                          ELSE IF .OPERATOR[TOKEN$B_KIND] EQL TOKEN$K_POSTFIX_OP
                                                DBG$PRINT(UPLIT BYTE(%ASCIC 'postfix'), 0)
                                                DBG$PRINT(UPLIT BYTE(%ASCIC 'invalid kind'), 0);
```

	PARS -000 843 844 845 846 847 848 849 850	ER		894 894 894 894 894 895	45678901			SPRI SNEW URN;								ffer, an	084 02:10 084 12:17 nd return		Page 284 (33)
22	43	41 64	21 28 6E	22 20 69	20 64 6B	20 72 65	79 6F 74 78 64	72 74 61 78 69	61 75 69 78 66 60	20 60 72 60 66 69 74 61	20 69 65 61 65 66 73 76	20 72 70 76 76 6F 6E	20 70 65 70 69 70 69 29	04 08 1A 20 06 05 07 00	03219 0321E 03227 03236 03242 0324F 03257 03264	P.AXN: P.AXO: P.AXP: P.AXR: P.AXS: P.AXT: P.AXU:	.ASCII .ASCII .ASCII .ASCII .ASCII .ASCII .ASCII	DBG\$PLIT,NOWRT, SHR, PIC,0 <4>\	
						7E		0	4	53 (6 63 63 63 63 63 63 63 63 63 63	00000	0000° 08 05 0E 04 29 04 30 04 36			00000 00002 00009 00010 00012 00014 00017 00028 00028 00028 00036 00036 00037 00046 00046 00046 00048 00046 00045 00055	DUMP_OP 18: 28: 38:	PSECT ERATOR: .WORD MOVAB MOVAB CLRL PUSHL CALLS BLBC CLRL PUSHAB CALLS ADDL3 PUSHAB CALLS CMPB BNEQ CLRL PUSHAB	DBG\$CODE,NOWRT, SHR, PIC,O Save R2,R3 DBG\$PRINT, R3 P.AXN, R2 -(SP) R2 W2, DBG\$PRINT PRIMARY_FLAG, 1\$ -(SP) P.AXO W2, DBG\$PRINT W12, OPERATOR, -(SP) P.AXP W2, DBG\$PRINT @OPERATOR, W2 2\$ -(SP) P.AXQ 5\$ @OPERATOR, W3 3\$ @OPERATOR, W4 4\$ -(SP) P.AXS 5\$ -(SP) P.AXS 5\$ -(SP)	8887 8921 8922 8924 8923 8924 8929 8931 8933 8935 8935 8937 8939

DBGPARSER V04-000				16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRCJDBGPARSER.B32;1	Page 285 (33)
0000000G	63 63 00	3E 4B	A2 7E A2 00	9F 00057 FB 0005A 5\$: CALLS #2, DBG\$PRINT D4 0005D CLRL -(SP) 9F 0005F PUSHAB P.AXU FB 00062 CALLS #2, DBG\$PRINT CALLS #2, DBG\$PRINT CALLS #0, DBG\$NEWLINE 04 0006C RET	8947 8948 8951

; Routine Size: 109 bytes, Routine Base: DBG\$CODE + 258D

Page 286 (34)

```
ROUTINE DUMP_TOKEN(TOKEN): NOVALUE =
                                FUNCTION
                                        This routine dumps out a specified Lexical Token Entry in a readable format. It prints one line of output of approximately this format:
                                                   token found: infix operator, code = 8, string = "**"
                                        This routine is used only for internal DEBUG debugging purposes. Its output is not seen by normal DEBUG users. The routine is only called by DBG$PRIMARY_PARSER if Developer Switch 2 is set.
                                INPUTS
                                        TOKEN
                                                   - The address of the Lexical Token Entry to dump.
                                OUTPUTS
                                        NONE
                                   BEGIN
                                        TOKEN: REF TOKENSENTRY;
                                                                                   ! Address of Lexical Token Entry to dump
                                  LOCAL KIND,
                                                                                      Pointer to text saying what kind of
                                                                                             lexical token this is
                                        NAMEPTR.
                                                                                      Pointer to the ASCIC name string for
                                                                                             the lexical token
                                                                                     Pointer to string saying whether this is a Primary Symbol operator
                                        PRIMARY;
                                     Determine what kind of Lexical Token Entry this is.
                                  CASE .TOKEN[TOKEN$B_KIND] FROM TOKEN$K_OPERAND TO TOKEN$K_POSTFIX_OP OF SET
                                        [TOKENSK OPERAND]:
                                             KIND = UPLIT BYTE(%ASCIC 'operand');
                                              NAMEPTR = TOKEN[TOKENSB_LENGTH];
                                        [TOKEN$K PREFIX_OP]:
    BEGIN
    KIND = UPLIT BYTE(%ASCIC 'prefix operator');
    NAMEPTR = TOKEN[TOKEN$B_OPLEN];
                                        [TOKENSK INFIX_OP]:
                                             KIND = UPLIT BYTE(%ASCIC 'infix operator');
```

```
N 3
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                                                                                                                                                                                                                                      VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32:1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  Page 287
(34)
       8909
8910
89112345
891145
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
891167
8911
                                                                                                                                            NAMEPTR = TOKEN[TOKEN$B_OPLEN];
                                                                                                                                            END:
                                                                                                                            [TOKENSK POSTFIX_OP]:
                                                                                                                                            KIND = UPLIT BYTE(%ASCIC 'postfix operator');
NAMEPTR = TOKEN[TOKEN$B_OPLEN];
                                                                                                                                            END:
                                                                                                                            [INRANGE, OUTRANGE]:
                                                                                                                                            KIND = UPLIT BYTE(%ASCIC 'invalid kind');
NAMEPTR = UPLIT BYTE(%ASCIC '');
                                                                                                                            TES:
                                                                                                                   Print the line describing the Lexical Token Entry.
                                                              9030
9031
9032
9033
9034
9035
                                                                                                            PRIMARY = UPLIT BYTE (%ASCIC '');

IF .TOKEN[TOKEN$V PRIMARY] THEN PRIMARY = UPLIT BYTE (%ASCIC ' (primary)');

DBG$PRINT(UPLIT BYTE (%ASCIC 'token found: !AC!AC, code = !SL, string = "!AC"'),

KIND, .PRIMARY, .TOKEN[TOKEN$W_CODE], .NAMEPTR);
                                                                                                              DBG$NEWLINE();
                                                              9036
9037
                                                                                                             RETURN:
       8937
       8938
                                                              9038
                                                                                                             END:
                                                                                                                                                                                                                                                                                                .PSECT
                                                                                                                                                                                                                                                                                                                               DBG$PLIT, NOWRT, SHR, PIC, 0
                                                                                                                                                                                                         6F
70
                                                                                                                                                                                                                                                               P.AXV:
                                                                                                                                                                                                                                                                                                .ASCII
                                                                                                                                                                                                                                                                                                                               <7>\operand\
                                                                                                                                                                                                                         072E0FC0000AF130
                                                                                                                                                                                                                                                               P.AXW:
                                                                                                                                                                                                                                                                                                                               <15>\prefix operator\
                                                                                                                                                                                                                                                                                                 .ASCII
                                                                                                                                                                                                         69
70
72
69
                                                                                                                                                                                          6E
6F
                                                                                                                                                                                                                                                                P.AXX:
                                                                                                                                                                          66
                                                                                                                                                                                                                                                                                                .ASCII
                                                                                                                                                                                                                                                                                                                               <14>\infix operator\
                                                                                                                                                                                                                                                                P.AXY:
                                                                                                                                                                                                                                                                                                 .ASCII
                                                                                                                                                                                                                                                                                                                               <16>\postfix operator\
                                                                                                                                                                          76
                                                                                                                                                                                                                                          0329E
032AB
                                                             69
                                                                                                                            69
                                              6E
                                                                              6B
                                                                                                                                                                                          6E
                                                                                                                                            60
                                                                                                                                                           61
                                                                                                                                                                                                                                                               P.AXZ:
                                                                                                                                                                                                                                                                                                                               <12>\invalid kind\
                                                                                                                                                                                                                                                                                                                              <0>
                                                                                                                                                                                                                                                                P.AYA:
                                                                                                                                                                                                                                                                                                 .ASCII
                                                                                                                                                                                                                                         032AC
032AD
032B8
032C7
032D6
032E0
                                                                                                                                                                                                                                                               P.AYB:
                                                                                                                                                                                                                                                                                                .ASCII
                                                              29
6E
65
                                                                              79
75
64
67
                                                                                             72
6F
6F
6E
                                                                                                             61
66
63
69
22
                                                                                                                                                                                                                                                                                                .ASCII
                                                                                                                                                                           70
68
41
20
22
                                                                                                                                                                                          28
6F
21
20
                                                                                                                                                                                                                                                                                                                               <10>\ (primary)\
                                                                                                                                            69
6E
274
41
                                                                                                                                                           72
65
43
73
21
                                                                                                                                                                                                          20
743
43
43
0
                                                                                                                                                                                                                                                                P.AYC:
                                                                                                                            6D
20
72
43
                                                                                                                                                                                                                                                                P.AYD:
                                                                                                                                                                                                                                                                                                 .ASCII
                                                                                                                                                                                                                                                                                                                               \/token found: !AC!AC, code = !SL, string\
                                                                                                                                                                                                                                                                                                 .ASCII \ = "!AC"\
                                                                                                                                                                                                                                                                                                .PSECT DBG$CODE,NOWRT, SHR, PIC,O
                                                                                                                                                                                                                      001C 00000 DUMP_TOKEN:
                                                                                                                                                                                                                                                                                                  WORD
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                8952
                                                                                                                                                                                                                                                                                                                               Save R2, R3, R4
```

P.AXZ, R4

MOVAB

54 00000000° EF 9E 00002

DBGPARSER V04-000						1	5-Sep- 4-Sep-	1984 02:10 1984 12:17	0:13 VAX-11 Bliss-32 V4.0-742 7:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 288 (34)
0027	003	51 01 001B	04	AC 61 0011	00 8f	00009 0000D		MOVL CASEB .WORD	TOKEN, R1 (R1), #1, #3 2\$-1\$,- 3\$-1\$,- 4\$-1\$,-	8988
		53 50	OD	64 A4	9E 9E	00019 00010 00020		MOVAB MOVAB	P.AXZ, KIND P.AYA, NAMEPTR	9022
		53	C8 08	1E A4 A1 A1	9E	00022	2\$:	MOVAB MOVAB BRB MOVAB MOVAB	P.AXV, KIND 8(R1), NAMEPTR	: 8988 : 8994 : 8995
		53	DO	14 0A	9E	0002A 0002C 00030	3\$:	MOVAB	P.AXW, KIND	; 8988 ; 9001
		53	E0	A4 04	9E	00032 00036	48:	MOVAB	6\$ P.AXX, KIND 6\$	9008
		53 50 52 04 52	0C 0E 01 0F	A4 A1 A4 A1 A5 A1 A5 A1 A5 A1 A5 A1 A5 A1 A5 A1 A5 A1 A5 A1 A5 A1 A1 A1 A1 A1 A1 A1 A1 A1 A1 A1 A1 A1	9E 9E 9E	00038	5\$: 6\$: 7\$:	BRB MOVAB BRB MOVAB MOVAB BLBC MOVAB PUSHL PUSHL PUSHL PUSHL PUSHL PUSHAB CALLS CALLS RET	P.AXY, KIND 12(R1), NAMEPTR P.AYB, PRIMARY 1(R1), 8\$ P.AYC, PRIMARY NAMEPTR 2(R1), -(SP) PRIMARY	9022 9023 8988 8994 8995 8988 9001 9002 9008 9009 9015 9016
		7E	02	A1	3C DD	0004E	8\$:	MOVZWL	NAMEPIR 2(R1), -(SP)	9034
	00000000G	00	1A	53 A4 05 00	9F FB 04	00040 00044 00048 0004C 00052 00054 00056 00060 00067		PUSHL PUSHAB CALLS CALLS RET	KIND P.AYD #5, DBG\$PRINT #0, DBG\$NEWLINE	9033 9035 9038

; Routine Size: 104 bytes, Routine Base: DBG\$CODE + 25FA

LENGTH = %UPVAL * (DBG\$K_PRIM_SIZE_ARRAY +

ELSE IF .NODEPTREDBG\$B_PNODE_FCODE] EQL RST\$K_TYPE_RECORD THEN

DBG\$K_PRIM_SIZE_SUB5*.NODEPTR[DBG\$B_PNARR_DIMCNT])

VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1

Page 289 (35)

```
DBGPARSER
V04-000
                                                                                                                        VAX-11 Bliss-32 V4.0-742
LDEBUG.SRCJDBGPARSER.B32;1
                                                                                       16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                        Page 290
(35)
  LENGTH = DBG$K_PRIM_SIZE_RECORD * %UPVAL
                                                 ELSE IF .NODEPTR[DBG$B_PNODE_FCODE] EQL RST$K_TYPE_VARIANT THEN
                                                      LENGTH = DBG$K_PRIM_SIZE_VARIANT * %UPVAL
                                                 ELSE
                                                      LENGTH = DBG$K_PRIM_SIZE_NORMAL * %UPVAL;
                                                DBG$DUMP_HEX(.NODEPTR, .LENGTH, .NODEPTR, UPLIT BYTE (%ASCIC ' Primary Sub-Node:'));
NODEPTR = .NODEPTR[DBG$L_PNODE_FLINK];
                                                 END:
                     9110
91112
91113
91114
91116
91121
91121
91121
91123
91123
91131
91131
91131
91131
91137
                                           END
                                        If this is a Value Descriptor, dump it as such.
                                      ELSE IF .PRIMPTR[DBG$B_DHDR_TYPE] EQL DBG$K_VALUE_DESC
                                      THEN
                                           DBG$DUMP_HEX(.PRIMPTR, .PRIMPTR[DBG$w_DHDR_LENGTH], .PRIMPTR, UPLIT BYTE (%ASCIC ' Value Descriptor:'))
                                        If this is a Volatile Value Descriptor, dump it as such.
                                      ELSE IF .PRIMPTR[DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC
                                      THEN
                                           DBG$DUMP_HEX(.PRIMPTR, .PRIMPTR[DBG$W_DHDR_LENGTH] .PRIMPTR,
UPLIT BYTE (%ASCIC ' Volatile Value Descriptor:'))
                                         If it is none of the above, something is wrong but we try to dump the
                                         descriptor anyway.
                                           We are all done--now return.
                                      RETURN;
                                      END:
                                                                                                     .PSECT
                                                                                                               DBG$PLIT, NOWRT, SHR, PIC, O
                           69
72
69
20
                                                 20
69
20
67
20
74
20
                                                                                          P.AYE:
                                                                                                     .ASCII
                                                                                                               <25>\
                                                                                                                              Primary Descriptor:\
           72
                                      50
74
50
65
65
72
56
                                            20
70
20
64
20
62
62
62
                                                      20
72
42
70
20
70
20
                                                            20
73
20
20
20
20
20
20
20
                                 72 6F 72 3A 61 3A
                                                                       20
65
20
75
20
63
20
                      60
                                                                                                     .ASCII
                                                                                                               <23>\
                                                                                                                              Primary Sub-Node:\
                                                                                         P.AYG:
                                                                                                     .ASCII
                                                                                                               <23>\
                                                                                                                              Value Descriptor:\
                                                                                                     .ASCII \
                                                                                                                          Volatile Value Descriptor:\
```

BGPARS	SER													1	5-Sep-19 4-Sep-19	84 02:10 84 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 29 (35
4 70 0 64 5 70	69 69 79	72 60 54	63 61 20	73 76 72	65 6E 6F	44 49 74	20 20 70	65 20 69	75 20 72	6C 20 63	61 3A 20 73	56 72 20 65	20 6F 1E 44 3A	03341 03350 03353 03362 03371	P.AYI:	.ASCII	<30>\ Invalid Descriptor Type:\	
																.PSECT	DBG\$CODE,NOWRT, SHR, PIC,0	
													007C		DUMP_PR	IMARY:	Save R2 R3 R4 R5 R6	; 903
									56 0	00000	0006	OO EF AC	9E	20000		MOVAB MOVAB	DBG\$DUMP_HEX, R6	
							7	9	54 8F	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	04	AC	90 91	00010		MOVL	Save R2,R3,R4,R5,R6 DBG\$DUMP_HEX, R6 P.AYE, R5 PRIMPTR, R4 2(R4), #121	907
												A4 51	12 BB	00019		MOVL CMPB BNEQ PUSHR PUSHL PUSHL CALLS	6\$ #^M <r4,r5></r4,r5>	907
												24	DD	00010		PUSHL	#36 84	
									66		14	04	FB DO	00021		CALLS	M4. DBG\$DUMP_HEX 20(R4), NODEPTR 20(R4), R0 NODEPTR, R0	908
									66 52 50 50		14	A4 52	9E	00028	15:	MOVL MOVAB CMPL	20(R4), RO NODEPTR, RO	908
									01		09	244450445045044504514	13	00021 00024 00028 0002C 0002F 00031		CMPL BEQL CMPB BNEQ MOVZBL	10\$ 9(NODEPTR), #1	908
									50		18	OD A2	12 9A	00035 00037 0003B 0003E 00042		BNEQ	2\$ 27(NODEPTR), RO	909
									50 50 53		28	14 A0 19	9E	0003B 0003E		MOVAB	#20, R0 40(R0), LENGTH	909
									07		09	A2	91	00044	2\$:	BRB CMPB BNEQ	9(NODEPTR), #7	909
									53			05 1C	00	00048 0004A		MOVL	3\$ #28, LENGTH	909
									13		09	A2	91	0004D 0004F	3\$:	MOVL BRB CMPB BNEQ MOVL BRB	9(NODEPTR), #19	909
									53			28	DO 11	00055		MONT	4\$ #40, LENGTH	910
									53			18	00 9F	00058 0005A	4\$: 5\$:	MOVL	#24, LENGTH	910
											1A	022583852C42C42C4C655	DD	00050	39:	MOVL PUSHAB PUSHL PUSHR CALLS MOVL	#24, LENGTH P.AYF NODEPTR #^M <r2,r3> #4, DBG\$DUMP_HEX (NODEPTR), NODEPTR</r2,r3>	910 910 910
									66			04	BB FB	00064		CALLS	#4, DBG\$DUMP_HEX	010
							,	A	8F		02	BC	D0	0006A	40.	BRB CMPB	1\$	910 908 911
							•	^	or		02	05	91 12 9F	00071	6\$:	BNEQ PUSHAB	1\$ 2(R4), #122 7\$:
								3	8F		32 02	OF	11 91	00048 0004A 0004F 00053 00058 00058 00062 00062 00067 00067 00078 00078 00078 00084 00087 00089	75:	BRB	P.AYG 9\$ 3(P4) #131	911 911 912
							٥	,	O.		4A	05 05 03	12 9F	0007D		BRB CMPB BNEQ PUSHAB	2(R4), #131 8\$ P.AYH	
											6B	03	11 9F	00082	ge.	BRB	93	912 912 913 913
									7E		00	A5 54	DD 3C	00087	8\$: 9\$:	BRB PUSHAB PUSHL MOVZWL	P.AYI R4 (R4), -(SP)	913

DBGPARSER V04-000

f 4 16-Sep-1984 02:10:13 VAX-11 BLiss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRCJDBGPARSER.B32;1

PUSHL R4 CALLS #4, DBG\$DUMP_HEX RET

9140

; Routine Size: 146 bytes, Routine Base: DBG\$CODE + 2662

```
ROUTINE FIX_UP_PRIMARY(PRIMPTR): NOVALUE =
                                               FUNCTION
                                                          This routine is needed to handle a special case that may arise in conjunction with the array slice feature: If X is a two - dimensional array, say 10x10, and the user specifies EX X[5], then we want to treat this as if he had said X[5][1:10]. However, we do not know at the time we are picking up the "5" subscript whether he is going to say X[5] or X[5][6], for example. So we cannot fix up the lower/upper bounds properly until after all the subscripts have been picked up. This is the routine that gets called after all the subscripts have been picked
                                                            that gets called after all the subscripts have been picked
                                                           up, to fix up these bounds.
                                                INPUTS
                                                           PRIMPTR - A pointer to the Primary Descriptor being constructed.
                                               OUTPUTS
                                                           The Primary Descriptor pointed to by PRIMPTR may be modified.
                                                   BEGIN
                                                           PRIMPTR: REF DBG$PRIMARY;
                                                   NODEPTR: REF DBG$PRIM_NOTE
                                                           SUBVECTOR: REF DBG$PRIM_NODE_SUBS;
                                                       Obtain a pointers to the Primary Descriptor Subnode and the
                                                       subscript vector within that subnode. Do nothing if the
                                                       subnode is not for an array.
                                                    NODEPTR = .PRIMPTR[DBG$L_PRIM_BLINK];
                                                    IF .NODEPTREDBGSB_PNODE_FCODES NEG RSTSK_TYPE_ARRAY
                                                           RETURN;
                                                   SUBVECTOR = NODEPTREDBGSA_PNARR_SVECTOR];
                                                       Check for the subscript count being less than the dimension count, but the range bit not being set. The point here is that we want to treat this case as if it really were a range (e.g., treat X[1] as being the same as X[1][1:10]. Note that this same code appears in GET_SUBSCRIPTS when we discover that we do have a range.
                                                    IF .NODEPTR[DBG$B_PNARR_SUBCNT] LSS .NODEPTR[DBG$B_PNARR_DIMCNT]
AND NOT .NODEPTR[DBG$V_PNARR_RANGE]
                                                    THEN
                                                          BEGIN
NODEPTR[DBG$V_PNARR_RANGE] = TRUE;
INCR I FROM 0 TO .NODEPTR[DBG$B_PNARR_SUBCNT] - 1 DO
                                                                   SUBVECTOR[.I, DBG$L PNSUB LBOUND] =
.SUBVECTOR[.I, DBG$L PNSUB SVALUE];
SUBVECTOR[.I, DBG$L PNSUB UBOUND] =
.SUBVECTOR[.I, DBG$L PNSUB_SVALUE];
```

DBGPARSER V04-000 : 9100 : 9101 : 9102 : 9103	9198 3 9199 2 9200 2 9201 1	END; RETURN; END;	ID;	H 4 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 294 (36)
		1B 29 OA 52	50 04 51 18 01 09 50 28 A1 18 A1 A1 A1 53 18 51 08 9E 00	39 12 0000E BNEQ 3\$ 8 A1 9E 00010 MOVAB 40(R1), SUBVECTOR F A1 91 00014 CMPB 31(NODEPTR), 27(NODEPTR) 2E 1E 00019 BGEQU 3\$ 03 E0 0001B BBS #3, 10(NODEPTR), 3\$ 08 88 00020 BISB2 #8, 10(NODEPTR)	9141 9174 9175 9178 9187 9188 9191 9192 9194 9195 9197

; Routine Size: 74 bytes. Routine Base: DBG\$CODE + 26F4

ROUTINE GET_BLISS_SUBSCRIPTS(PRIMPTR, NAME): NOVALUE = **FUNCTION**

This routine picks up subscript values in a BLISS structure reference (i.e., any BLISS primary of the form X[...]). It calls DBG\$EXPRESSION_PARSER to pick up each subscript.

The kinds of BLISS structures we accept are:

bitvector vector X[n,p,s,e] block X[m,n,p,s,e] blockvector

This routine assumes that the opening subscript bracket has already been found and that the parse pointer points to the start of the first subscript expression. When this routine returns, the parse pointer is left pointing at the first character after the closing subscript bracket.

INPUTS

PRIMPTR -A pointer to the Primary Descriptor for a structure about

to be subscripted. NAME - The name of the object being subscripted (for error message purposes)

OUTPUTS The PRIMPTR Primary Descriptor is changed to include the subscript information. This is represented as follows:

X[i] bitvector - i is stored in a Primary Descriptor Array

sub-node.
i is stored in a Primary Descriptor Array X[i] vector sub-node.

X[n,p,s,e] block - n is stored in a Primary Descriptor Array

sub-node. p, s, and e are stored in the Primary Descriptor Root node, in the prim offset, prim length, and sgnext fields. m and n are stored in a Primary Descriptor Array sub-node. p, s, and e are stored in the Primary Descriptor Root node, in the prim offset, prim length, and sgnext fields. X[m,n,p,s,e] blockvector-

BEGIN

PRIMPTR: REF DBG\$PRIMARY;

Pointer to BLISS structure Primary Descriptor.

LOCAL

DECLTYPE: REF DBG\$VALDESC.

DSTPTR: REF DST\$RECORD, FCODE, LA PTR: REF VECTOR[, BYTE], LOW_RANGE_VAL,

Count of field values Pointer to value descriptor for the subscript value.
Pointer to BLISS structure DST entry
Data type FCODE for current symbol Lookahead pointer into input The lower value of a subscript range

```
DBGPARSER
V04-000
                                                                                                                                        VAX-11 Bliss-32 V4.0-742
LDEBUG.SRCJDBGPARSER.B32:1
                                                                                                   16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                 NODEPTR: REF DBG$PRIM_NODE, ! Pointer to Prim Desc Array sub-node NODESUBPTR: REF DBG$PRIM_NODE_SUBS, ! Pointer to subscript blockvector
  in Prim Desc Array sub-node
                                                                                                      Temporary pointer to field name values
True for REF objects
Pointer to BLISS structure RST entry
Temporarily saved expression radix
                                                 PTR: REF VECTOR[,LONG],
                                                 REF FLAG,
RSTPTR: REF RSTSENTRY,
                                                  SAVED RADIX,
                                                                                                      Stride for blocks and blockvectors
Code for kind of BLISS structure
                                                 STRUC,
SUBSCR_COUNT,
SUBVECTOR: VÉCTOR[5],
                                                                                                      Count of the number of subscripts
Vector of subscripts
                                                 TOKEN,
TYPEID,
                                                                                                      Pointer to a Lexical Token
                                                                                                      Pointer to a typeid
                                                 VALPTR: REF DBG$VALDESC,
                                                                                                      Pointer to subscript Value Descriptor
                                                  VALUE:
                                                                                                      Subscript value
                                           DBG$GL_CURRENT_PRIMARY = .PRIMPTR;
                                              Check that the Primary Descriptor has the correct Kind and FCODE.
                                              If so, obtain a pointer to the DST record for this BLISS structure.
                                           RSTPTR = .PRIMPTR [DBG$L_DHDR_SYMIDO];
                                           IF .RSTPTR EQL O
                                           SIGNAL (DBG$ NOTASTRUCT, 1, .NAME);
DSTPTR = .RSTPTR [RST$L_DSTPTR];
IF .PRIMPTR [DBG$B_DHDR_KIND] NEQ RST$K_DATA
THEN
                                                 SIGNAL (DBG$_NOTASTRUCT, 1, DSTPTR[DST$B_NAME]);
                                           DBG$STA_SYMTYPE (.RSTPTR, FCODE, TYPEID);
                                           IF .FCODE NEG RSTSK_TYPE_BLIDATA
                                                 SIGNAL (DBG$_NOTASTRUCT, 1, DSTPTR[DST$B_NAME]);
                                              Check for no field structure in the RST - this arises when the user defines his own BLISS structure, instead of using one of the builtins BITVECTOR, VECTOR, BLOCK, or BLOCKVECTOR. We do
                                              not handle this case.
                                           STRUC = .DSTPTR [DST$V_BLI_STRUC];
IF .STRUC EQL DST$K_BLI_NOSTRUC
                                            THEN
                                                 SIGNAL (DBG$_NOTASTRUCT, 1, DSTPTR[DST$B_NAME]);
                                              If the REF bit is set, then there is a sub-node for the dereferencing. Call DBG$BUILD_PRIMARY_SUBNODE to build a new subnode for the array information. Light the EVAL bit in the dereference subnode.
                                            IF .DSTPTR [DST$V_BLI_REF]
                                            THEN
                                                 DBG$BUILD_PRIMARY_SUBNODE (.PRIMPTR, RST$K_DATA, O, RST$K_TYPE_BLIDATA, .PRIMPTR [DBG$L_DHDR_TYPEID], .DSTPTR);
                                                 NODEPTR = .PRIMPTR [DBG$L_PRIM_FLINK];
```

TOKEN = DBG\$LEXICAL_SCANNER (FALSE, FALSE, SUBSCRIPT_TERM_TBL, 0);

IF .TOKEN NEQ TERMINATOR_TOKEN
THEN

BEGIN

```
M 4
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                                                    VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                                                          COUNT = .PTR[0];
INCR I FROM 1 TO .COUNT DO
BEGIN
IF .SUBSCR_COUNT LSS 5 THEN SUBVECTOR[.SUBSCR_COUNT] = .PTR[.I];
SUBSCR_COUNT = .SUBSCR_COUNT + 1;
                                                                           END
                                                                      The subscript is not a BLISS field name so we pick up the subscript value and convert it to integer. This subscript value is then put away in the subscript vector and the subscript count is incremented.
                                         ELSE
                                                                           BEGIN
                                                                              Convert the value descriptor to an integer value.
                                                                           VALUE = CONVERT_TO_INTEGER (.VALPTR);
                                                                              If the terminator is a colon, we have the first part of a sub-
script range specification (such as 'ARR[2:5]'). See if a range
                                                                              specification is allowed (it is only allowed on the first sub-
                                                                              script of a bitvector, a normal vector, or a blockvector) and save away this lower value of the range.
                                                                           IF .TERMINATOR_CODE EQL TOKEN$K_TERM_COLON THEN
                                                                                  BEGIN
                                                                                 IF .NODEPTR[DBG$V_PNARR_RANGE] OR
(.SUBSCR_COUNT_NEQ_O) OR
((.STRUC_NEQ_DST$K_BLI_BITVEC) AND
(.STRUC_NEQ_DST$K_BLI_VEC) AND
(.STRUC_NEQ_DST$K_BLI_BLKVEC))
                                                                                  THEN
                                                                                          SIGNAL (DBG$_INVRANSPEC);
                                                                                 NODEPTR[DBG$V_PNARR_RANGE] = TRUE;
LOW_RANGE_VAL = .VALUE;
END
                                                                              The terminator is not a colon, so we have the complete specifica-
                                                                              tion for the current subscript. Here we simply fill the subscript value (or the upper bound of the range) into SUBVECTOR and increment the subscript count. We can ignore subscripts after the fifth one because they will be in error anyway.
                                                                          ELSE
                                                                                  BEGIN
                                                                                  IF .SUBSCR_COUNT LSS 5
                              9483
9484
9485
9486
                                                                                          SUBVECTOR[.SUBSCR_COUNT] = .VALUE;
```

Page 299 (37)

Page 300 (37)

fill in the Primary Descriptor Array Sub-Node.

NODEPTR [DBG\$B_PNARR_SUBCNT] = 1; NODESUBPTR [0, DBG\$L_PNSUB_SVALUE] = .SUBVECTOR[0];

SIGNAL (DBG\$_ILLOFFSET, 1, .SUBVECTOR[1]);

THEN

9560

```
5
DBGPARSER
V04-000
                                                                                                                                                                    VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
                                                                                                                        16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                   IF (.SUBVECTOR[2] LSS 0)
THEN
   SIGNAL (DBG$_ILLLENGTH, 1, .SUBVECTOR[2]);
                                                                   IF (.SUBVECTOR[2] GTR 32)
THEN
                                                                           BEGIN
                                                                           SUBVECTOR[2] = 32;
SIGNAL(DBG$_SIZETRUNC);
                                                                    IF (.SUBVECTOR[3] NEQ 0) AND (.SUBVECTOR[3] NEQ 1)
                                                                   THEN
                                                                           SIGNAL(DBG$_ILLSIGEXT, 1, .SUBVECTOR[3]);
                                                                    ! Fill in the Primary Descriptor Sub-Node.
                                                                  PRIMPTR [DBG$V_DHDR_BLIBLK] = TRUE;
PRIMPTR [DBG$V_DHDR_SUBREF] = TRUE;
PRIMPTR [DBG$V_DHDR_BITREF] = TRUE;
PRIMPTR [DBG$W_PRIM_OFFSET] = .SUBVECTOR[1];
PRIMPTR [DBG$W_PRIM_LENGTH] = .SUBVECTOR[2];
PRIMPTR [DBG$V_DHDR_SGNEXT] = .SUBVECTOR[3];
NODEPTR [DBG$B_PNARR_SUBCNT] = 1;
NODESUBPTR [O, DBG$L_PNSUB_SVALUE] = .SUBVECTOR[0];
                                                                   END:
                                                                                                                        ! End of BLISS block case
                                                               Handle the Blockvector case.
                                                           [DST$K_BLI_BLKVEC]:
                                                                      We previously represented the block as a block of longwords,
                                                                      for purposes of aggregate output. If we get here, however, we are no longer doing aggregate output. So, fix up the information here, filling in the correct stride.
                                                                  STRIDE = .DSTPTR [DST$B_BLI_BLKVEC_UNIT_SIZE];
NODESUBPTR [1, DBG$L_PNSUB_STRIDE] = .STRIDE;
NODESUBPTR [1, DBG$L_PNSUB_UBOUND] =
.DSTPTR [DST$L_BEI_BLKVEC_UNITS]-1;
NODEPTR[DBG$W_PNARR_LENGTH] = .STRIDE;
IF .STRIDE EQE 1
THEN
                                                                          IF .SUBVECTOR[4]
                                                                  NODEPTR [DBG$B_PNARR_DTYPE] = DSC$K_DTYPE_BU
THEN
                                                                                  NODEPTR [DBG$B_PNARR_DTYPE] = DSC$K_DTYPE_B
                                                                          IF .SUBVECTOR[4]
```

```
DBGPARSER
V04-000
                                                                            16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                         VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
  ELSE IF .STRIDE EQL 4
                                                    NODEPTR [DBG$B_PNARR_DTYPE] = DSC$K_DTYPE_W
                                                IF .SUBVECTOR[4]
                                                THEN
                                                    NODEPTR [DBG$B_PNARR_DTYPE] = DSC$K_DTYPE_L
                                                    NODEPTR [DBG$B_PNARR_DTYPE] = DSC$K_DTYPE_LU
                                           ELSE
                                               BEGIN
IF .SUBVECTOR[4]
                                                THEN
                                                    NODEPTR [DBG$B_PNARR_DTYPE] = DSC$K_DTYPE_V
                                                    NODEPTR [DBG$B_PNARR_DTYPE] = DSC$K_DTYPE_VU;
                                               NODEPTREDBGSW_PNARR_LENGTH] = 8 * .NODEPTREDBGSW_PNARR_LENGTH];
                                             Check that the block-vector had exactly five subscripts.
                                           IF .SUBSCR_COUNT LSS 5 THEN SIGNAL(DBG$_TOOFEWSUB, 1, 5);
IF .SUBSCR_COUNT GTR 5 THEN SIGNAL(DBG$_TOOMANSUB, 1, 5);
                                             Check that the subscript values are in range.
                                           IF NOT .REF_FLAG
                                           THEN
                                               BEGIN
                                                   (.SUBVECTOR[O] LSS 0) OR (.SUBVECTOR[O] GEQ .DSTPTR[DST$L_BLI_BLKVEC_BLOCKS])
                                                    SIGNAL (DBG$_STRUCSIZE, 2,
                                                            .DSTPTREDST$L_BLI_BLKVEC_BLOCKS], .SUBVECTOR[0]);
                                               IF (.SUBVECTOR[1] LSS 0) OR (.SUBVECTOR[1] GEQ .DSTPTR[DST$L_BLI_BLKVEC_UNITS])
                                               THEN
                                                    SIGNAL (DBG$_STRUCSIZE, 2, .DSTPTR[DST$L_BLI_BLKVEC_UNITS], .SUBVECTOR[1]);
                                               END:
                                           IF (.SUBVECTOR[2] LSS -%x'8000') OR (.SUBVECTOR[2] GTR %x'7FFF')
                                           THEN
                                               SIGNAL(DBG$_ILLOFFSET, 1, .SUBVECTOR[2]);
                                           IF (.SUBVECTOR[3] LSS 0)
                   9765
                                           THEN
                                               SIGNAL(DBG$_ILLLENGTH, 1, .SUBVECTOR[3]);
                   9767
                   9768
9769
                                           IF (.SUBVECTOR[3] GTR 32)
                                           THEN
                                                SUBVECTOR[3] = 32;
```

```
DBGPARSER
V04-000
                                                                                                                                   16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                                    VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGPARSER.B32:1
                                                                                                                                                                                                                                                              Page 306
(37)
                                                         IF .NODEPTR[DBG$V_PNARR_RANGE]
THEN
   BEGIN

IF .LOW_RANGE_VAL GTR .SUBVECTOR[0] THEN SIGNAL(DBG$_INVRANSPEC);

NODESUBPTR[0, DBG$L_PNSUB_SVALUE] = .LOW_RANGE_VAL;

NODESUBPTR[0, DBG$L_PNSUB_LBOUND] = .LOW_RANGE_VAL;

NODESUBPTR[0, DBG$L_PNSUB_UBOUND] = .SUBVECTOR[0];
                                                                  RETURN:
                                                                 END:
                                                             Build a new subnode. The typeid that we pass in to BUILD_PRIMARY_SUBNODE describes the element referenced by the subscript expression. This typeid is pulled from the CELLTYPE field.
                                                         DBG$BUILD_PRIMARY_SUBNODE (.PRIMPTR, RST$K_DATA, 0, RST$K_TYPE_ATOMIC, .NODEPTREDBG$L_PNARR_CELLTYPE], 0);
                                                          RETURN:
                                                         END:
                                                                                                                OFFC 00000 GET_BLISS_SUBSCRIPTS:
                                                                                                                                                                       Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
#60, SP
PRIMPTR, R6
R6, DBG$GL_CURRENT_PRIMARY
12(R6), RSTPTR
                                                                                                                                                                                                                                                                      9202
                                                                                                                          00002
00005
00009
00010
00014
00016
                                                                                                                    00
                                                                                                                                                        SUBL 2
                                                                                                            3A5662C1F32A682A1F38E23E2A1F30
                                                                                                                                                                                                                                                                      9275
                                                                                                                                                        MOVL
                                                                                                                    0000000G
                                                                                                                                                        MOVL
                                                                                                                                                                                                                                                                      9280
9281
9283
                                                                                                  00
                                                                                                                                                        MOVL
                                                                                                                                                       BNEQ
                                                                                                  08
                                                                                                                                                        PUSHL
                                                                                                                                                                        NAME
                                                                                                                          00019
0001B
                                                                                                                                                       PUSHL
                                                                                                                                                                       #164264
#3, LIB$SIGNAL
12(RSTPTR), DSTPTR
4(R6), 4(SP)
#24, #8, a4(SP), #6
                                                                                      000281A8
                                                                                                                                                        PUSHL
                                                                                                                          00021
00028
00021
00031
00037
00039
00036
00044
00048
00048
00051
00053
00058
00065
00065
00065
00065
00065
00065
00065
                                                                                                                    FD9ED3FDDBFFFDDBFFF2
                                                         0000000G
                                                                                                                                                        CALLS
                                                                                                                                                                                                                                                                      9284
9285
                                                                                                  00
                                                                                                                                                        MOVL
                                                                                AE
08
                                                                                                                                                        MOVAB
                                                                      04
                                                                                                                                                        CMPZV
                      06
                                                                                                                                                        BEQL
                                                                                                                                                        PUSHAB
                                                                                                                                                                        7(DSTPTR)
                                                                                                                                                                                                                                                                      9287
                                                                                                  07
                                                                                                                                                        PUSHL
                                                                                      000281A8
                                                                                                                                                        PUSHL
                                                                                                                                                                        #164264
                                                                                                                                                                       #3, LIB$SIGNAL
                                                                                                                                                       CALLS
PUSHAB
                                                          0000000G
                                                                                00
                                                                                                                                                                                                                                                                      9289
                                                                                                  10
                                                                                                                                                        PUSHAB
                                                                                                                                                                        FCODE
                                                                                                                                                       PUSHL
CALLS
CMPL
                                                                                                                                                                        RSTPTR
                                                                                                                                                                        #3. DBG$STA_SYMTYPE
ECODE, #13
                                                          0000000G
                                                                                                                                                                                                                                                                      9290
                                                                                                  20
                                                                                                                                                        BEQL
                                                                                                                                                       PUSHAB
                                                                                                                                                                                                                                                                     9292
                                                                                                  07
                                                                                                                                                                        7(DSTPTR)
                                                                                                                                                        PUSHL
                                                                                                                                                                       #164264
#3, LIB$SIGNAL
#0, #3, 5(DSTPTR), STRUC
                                                                                                                                                       PUSHL
CALLS
EXTZV
BNEQ
PUSHAB
                                                          0000000G
                                                                                                                                                                                                                                                                     9300
9301
9303
                       58
                                                                                                                                                                       7(DSTPTR)
                                                                                                  07
```

				H 5 16-Sep- 14-Sep-	-1984 02:10 -1984 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 307 (37)
0000000G	00	000281A8 05	01 I 8F I 03 I A4	DD 0007D DD 0007F FB 00085 95 0008C 4\$: 18 0008F DD 00091 DD 00098 DD 00098 DD 00098 DD 0009B FB 0009D DO 000AZ 88 000A6 DO 000AA	PUSHL PUSHL CALLS TSTB BGEQ PUSHL PUSHL MOVQ PUSHL CALLS MOVL BISB2	#1 #164264 #3, LIB\$SIGNAL 5(DSTPTR)	9310
		08	54 I	DD 00091 DD 00093 DD 00096	PUSHL PUSHL PUSHL	DSTPTR 8(R6) #13	9314
D88C	7E		56	7D 00098 DD 0009B FB 0009D	PUSHL	#6, -(SP) R6 M6 DRGSBULL D DRIMARY SUBNODE	
0A	CF 52 A2 6E	14	A6 01 01 02	DO 000A2 88 000A6 DO 000AA 11 000AD	MOVL BISB2 MOVL BRB	#6, DBG\$BUILD_PRIMARY_SUBNODE 20(R6), NODEPTR #1, 10(NODEPTR) #1, REF_FLAG 6\$	9315 9316 9317 9310
02	52 53 58 AB	18 28 08		DO 000A2 88 000A6 DO 000AA 11 000AD D4 000AF 9E 000B5 9E 000B9 88 000BD D4 000C1 DO 000C3 D1 000CA 7\$: 12 000D1 31 000D3 D1 000D0 95: 12 000E0 D6 000E2 11 000E4	MOVL BRB CLRL MOVAB MOVAB BISB2 CLRL MOVL CMPL BNEQ BRW MOVL CMPB BNEQ	REF_FLAG 24(R6), NODEPTR 40(R2), NODESUBPTR 8(NODEPTR), R11 #1, 2(R11) SUBSCR_COUNT #1, TERMINATOR_CODE TERMINATOR_CODE, #2 8\$	9315 9316 9317 9310 9320 9326 9327 9328
00000000.	EF 02	00000000	55 I	D4 000C1 D0 000C3 D1 000CA 7\$: 12 000D1 31 000D3	CLRL MOVL CMPL BNEQ	SUBSCR COUNT #1, TERMINATOR CODE TERMINATOR_CODE, #2 8\$	9336 9337 9338
	59 20	00000000.01	69	31 00003 D0 00006 8\$: 91 00000 9\$: 12 000E0 D6 000E2 11 000E4	BRW MOVL CMPB BNEQ	8\$ 29\$ CHARPTR, LA PTR (LA PTR), #32 10\$	9346 9347
	2A		59 69 03 03 13	D6 000E2 11 000E4 91 000E6 10\$: 13 000E9 31 000EB E0 000EE 11\$:	INCL BRB CMPB BEQL BRW	LA_PTR 9\$ (LA_PTR), #42 11\$	9348
13	6B	00	13 6	EO 000EE 11\$:	DDC	17\$ #19, (R11), 12\$ SUBSCR_COUNT 12\$	9351 9352
	02		55 0F 58 17	05 000F2 12 000F4 01 000F6 13 000F9	TSTL BNEQ CMPL BEQL CMPL BEQL CMPL BEQL PUSHL	STRUC, #2 13\$	9353
	01		58 [01 000FB	CMPL	STRUC, #1	9354
	04		58 I	01 00100 13 00103	CMPL BEQL	STRUC. #4	9355
00000000g	00 EF	00028F08	8F (DD 00105 12\$: FB 0010B	PUSHL CALLS MOVAB	13\$ #167688 #1,_LIB\$SIGNAL	9357
00000000	EF	00000000	58 1 58 0 8F 1 89 1 7E 1 7E 1	05 000F2 12 000F6 13 000F9 01 000FB 13 000FE 01 00100 13 00103 13 00103 14 00112 15 00114 16 00116 17 00126 18 00126 19 00127 19 00136 10 00136 11 00136 12 00136 13 00136 13 00136 13 00136 14 00140 15 00140 16 00140	MOVAB INCL CLRL PUSHL	#1, LIB\$SIGNAL 1(R9), CHARPTR SUBSCR_COUNT -(SP) SUBSCRIPT_TERM_TBL -(SP) #4, DBG\$LEXICAL_SCANNER	9359 9360 9368 9369 9368
E235	CF AE 50 50	00000000.	7E 04 50 EF AE 1E 01 FF	7C 00124 FB 00126 DO 0012B	INCL CLRL PUSHL CLRQ CALLS MOVL MOVAB	-(SP) #4, DBG\$LEXICAL_SCANNER RO, TOKEN TERMINATOR_TOKEN, RO	9368
	50	18	AE I	01 00136	CMPL	TOKEN, RO	1 7370
24	AE	00000000	O1 S	90 0013C 90 00140 9F 00148	CMPL BEQL MOVB MOVB PUSHAB	#1, ASCIC_STRING acharptr, ASCIC_STRING+1 ASCIC_STRING	9375 9376 9377

					16	-Sep-198	4 02:10	13	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page 308 (37)
	00000000G	00	000289E2	01 DD 001 8F DD 001 03 FB 001 EF D1 001 0D 12 001 8F DD 001 01 FB 001 EF D5 001	4803A		PUSHL PUSHL CALLS CMPL BNEQ	TERMIN	0 B\$SIGNAL ATOR_CODE, #3	9379
	00000000G	00	00028F08	01 DD 001 8F DD 001 EF D1 001 00 12 001 8F DD 001 01 FB 001 EF D5 001 00 12 001 8F DD 001	63		PUSHL	15\$ #16768 #1, LI TERMIN	8 B\$SIGNAL ATOR_CODE	9381 9382
	000000006 000000000 02 28	OO EF AB 5A AE	00000000°	8F DD 001 01 FB 001 EF CO 001 08 88 001 A3 DO 001 A3 DO 001 F2A 31 001	78 7E 85 90 94	16\$:	TSTL BNEQ PUSHL CALLS ADDL2 BISB2 MOVL MOVL	TERMIN	8 B\$SIGNAL ATOR_LENGTH, CHARPTR R11) SUBPTR), LOW_RANGE_VAL ESUBPTR), SUBVECTOR	9384 9385 9390 9391 9392 9348
	000000000	AE EF	00000000	OA DO 001	AO AB AF	17\$:	BRW MOVL MOVL PUSHL	/\$	SION_RADIX, SAVED_RADIX XPRESSION_RADIX IPT_TERM_TBL	9348 9404 9405 9406
	DCFE	CF 57 EF	000000000	7E D4 001 02 FB 001 50 D0 001 AE D0 001 EF D5 001 0D 12 001	BC BF C7		CLRL CALLS MOVL MOVL TSTL	RO, VA SAVED TERMIN	G\$EXPRESSION_PARSER LPTR RADIX, EXPRESSION_RADIX ATOR_CODE	9407 9414
	000000000	00 EF OE	00000000° 00028E90	FF DD 001 7E D4 001 50 D0 001 FB 001 FF D5 001 FF D5 001 FF D0 001	CF D5 DC E7	18\$:	PUSHL CALLS ADDL2 CMPB	#16756 #1, LI	8 B\$SIGNAL ATOR_LENGTH, CHARPTR TR), #14	9415 9422
11		6B	A	13 E1 001 55 D5 001 00 12 001	F1 F3		BNEQ BBC TSTL BNEQ	#19, (SUBSCR 19\$	R11), 19\$ _COUNT	9425
	00000000G 08 14	00 AE AE	00028F08 20 08		F 5 F B	19\$:	PUSHL CALLS MOVAB	#16768 #1, LI 32(R7)	8 B\$SIGNAL PTR COUNT	9427 9429 9430 9431
		05	06	01 FB 001 A7 9E 002 BE D0 002 50 D4 002 0E 11 002 55 D1 002 07 18 002	0C 0E 10	19\$: 20\$:	MOVAB MOVL CLRL BRB CMPL	1	_COUNT, #5	9431
ED	28 A		08 BI	55 D6 002 AE F3 002	15 1C 1E	21\$: 22\$:	BRB CMPL BGEQ MOVL INCL AOBLEQ BRB PUSHL CALLS MOVL CMPL BNEQ BBS TSTL BNEQ CMPL	PTR[I	J, SUBVECTOR[SUBSCR_COUNT] _COUNT _I, 20\$	9434 9431 9422 9450
	FB33 00	CF AE 03	00000000.	57 DD 002	25 27 20 30	23\$:	PUSHL CALLS MOVL CMPL	VALPTR #1, CO RO, VA TERMIN	NVERT_TO_INTEGER LUE ATOR_CODE, #3 R11), 24\$	9450
13		6B		2E 12 002 13 E0 002 55 D5 002 0F 12 002	37 39 30		BNEQ BBS TSTL RNEQ	26\$ #19, (SUBSCR	R11), 24\$ _COUNT	9462 9463
		02		01 FB 002 50 D0 002 EF D1 002 13 E0 002 55 D5 002 0F 12 002 58 D1 002 17 13 002 18 D1 002	41 44 46 49		CMPL BEQL CMPL BEQL	STRUC, 25\$ STRUC, 25\$	#2	9464

DBGPARSER V04-000	J 5 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 309 (37)
00A9	04 58 D1 0024B CMPL STRUC, #4 00 13 0024E 24\$: PUSHL #167688 00000000G 00 01 FB 00256 24\$: PUSHL #167688 02 AB 08 88 00250 25\$: BISB2 #8, 2(R11) 5A 0C AE D0 00261 MOVL VALUE, LOW_RANGE_VAL 05 55 D1 00267 26\$: CMPL SUBSCR_COUNT, #5 06 18 0026A BGEQ 27\$ 28 AE45 0C AE D0 0026C MOVL VALUE, SUBVECTOR[SUBSCR_COUNT] 55 D6 00272 27\$: INCL SUBSCR_COUNT 65 TE53 31 00274 28\$: BRW 7\$ 04 00 58 CF 00277 29\$: CASEL STRUC, #0, #4 001A 0036 000A 0027B 30\$: WORD 31\$-30\$,-	9466 9468 9470 9471 9459 9483 9485 9487 9338 9500
	00000000 00 002832A 8F DD 00285 31\$: PUSHL #164650 00000000 00 01 FB 0028B CALLS #1, LIB\$SIGNAL 037E 31 00292 BRW 85\$ 55 D5 00295 32\$: TSTL SUBSCR_COUNT 11 14 00297 BGTR 33\$	9821 9512
	1C 14 002AD BGTR 36\$ 2B 11 002AF BRB 37\$ 55 D5 002B1 34\$: TSTL SUBSCR_COUNT 11 14 002B3 BGTR 35\$ 01 DD 002B5 PUSHL #1 01 DD 002B7 PUSHL #1	9513 9518 9556
	00000000G 00 03 FB 002BF 35\$: CMPL SUBSCR_COUNT, #1 11 15 002C9 BLEQ 37\$ BLEQ 37\$ 01 DD 002CB 36\$: PUSHL #1 01 DD 002CB PUSHL #1 0000000G 00 03 FB 002D5 CALLS #3, LIB\$SIGNAL SUBVECTOR, R0 05 DD 002EB 00 DD 002EB BLSS 38\$ CMPL R0, 6(DSTPTR) BLSS 39\$ 0000000G 00 0000000G 00 0000000 DD 0000000 DD 0000000 DD 000000	9557 9562 9565
	000000000	9566 9569 9568 9575 9576 9577

R						1	S-Sep-	1984 02:10 1984 12:17	0:13 VAX-11 Bliss-32 V4.0-742 7:30 CDEBUG.SRCJDBGPARSER.B32;1	Page 310 (37)
		000000006		02 8F 04 0143	DD DD DD FB 31	0030D 0030F 00312 00314 0031A 00321 00324	40\$: 41\$: 42\$:	PUSHL PUSHL PUSHL CALLS BRW EXTZV	LOW_RANGE_VAL 6(DSTPTR) #2 #163963 #4. LIB\$SIGNAL 61\$	9580 9579 9586
50	0A 0C	A4 A3 06 10	04 A3 A4 A2 51 18	00 50 01 50 8 82 50	EF DO C3 B0 PE D1	00324 0032E 00334 00338 0033C 0033F	425:	EXTZV MOVL SUBL3 MOVW MOVAB CMPL BNEQ BLBC MOVB	#0, #4, 10(DSTPTR), STRIDE STRIDE, 4(NODESUBPTR) #1, 6(DSTPTR), 12(NODESUBPTR) STRIDE, 28(NODEPTR) 24(NODEPTR), R1 STRIDE, #1	9586 9601 9602 9604 9605 9610 9606
		02	06 34 A1		E 9	00341 00345 00349		BLBC MOVB	SUBVECTOR+12, 43\$ #6, 2(R1) 51\$	9608
		02	A1 02	AE 062 022 030 050 10E 07	90 11 D1	0034B	43\$: 44\$:	MOVB	#2, 2(R1) 51\$	9612 9608 9613
		02	06 34 A1		12 E9 90	00351 00354 00356 0035A 00360 00364	****	CMPL BNEQ BLBC MOVB	STRIDE, #2 46\$ SUBVECTOR+12, 45\$ #7, 2(R1) 51\$	9615 9617
		02	A1 04	2D 03 27 50	90 11 D1	0035E 00360 00364 00366	45\$: 46\$:	MOVB BRB	515 #3, 2(R1) 515 STRIDE, #4	9619 9615 9620
		02	06 34 A1	10	12 E9	00366 00369 0036B 0036F		CMPL BNEQ BLBC MOVB	48\$ SUBVECTOR+12, 47\$ #8, 2(R1) 51\$	9622 9624
		02	A1	04	90	00375	47\$:	BRB MOVB BRB	#4, 2(R1) 51\$	9626
		02	06 34 A1	04 12 AE 01 04	90 11	0036F 00373 00375 00379 0037B 00383	48\$:	BLBC MOVB BRB	SUBVECTOR+12, 49\$	9626 9622 9629 9631
		02 1C	A1 A2 04	22 08 55 11	90 A4 D1 18 DD	00385 00389 00380 00390 00392	49\$: 50\$: 51\$:	MOVB MULW2 CMPL BGEQ PUSHL	#34, 2(R1) #8, 28(NODEPTR) SUBSCR_COUNT, #4 52\$ #4	9633 9634 9639
		000000006	00028EA0	03 55 11	90418DDDDBB15DDDDBB8091DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD	00385 00389 00389 00390 00394 00396 00396 00388 00388 00388 00388 00386 00386 00386 00386 00386 00386 00386 00386	52\$:	MOVB MULW2 CMPL BGEQ PUSHL PUSHL CMPL PUSHL	#1 #167584 #3, LIB\$SIGNAL SUBSCR_COUNT, #4 53\$ #4	9640
		0000000G	00028EB0	03 6E	DD FB E8	003AA 003AC 003B2 003B9 003BC	53\$:	PUSHL PUSHL CALLS BLBS MOVL	#167600 #3, LIB\$SIGNAL REF FLAG, 55\$ SUBVECTOR, RO 54\$ RO, 6(DSTPTR) 55\$ RO 6(DSTPTR)	9645 9647
		06	A4	06 50	19	003C0 003C2		BLSS	RO, 6(DSTPTR)	9648
			06	50	19 00 00 00	003C8 003CA 003CD	54\$:	BLSS PUSHL PUSHL PUSHL	RO 6(DSTPTR)	9651 9650

DBGPARSER V04-000							1	-Sep-	1984 02:10 1984 12:17	:13	VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32;1	Page 311 (37)
		00000000G FFFF8000	00 8F	0002807B 2C	8F 04 AF	DD FB D1	003CF 003D5 003DC	558:	PUSHL CALLS CMPL	#163 #4 SURV	3963 LIB\$SIGNAL VECTOR+4, #-32768	9653
		00007FFF	8F	20	AE OA AE	19	003E4 003E6		BLSS	56\$ SUBV	VECTOR+4, #32767	9654
				20	16	15 DD	003EE	56\$:	CMPL BLSS CMPL BLEQ PUSHL PUSHL PUSHL CALLS	57\$	VECTOR+4	9656
		00000000	00	000280F0	AE 01 8F 03	DD DD FB	003F3		PUSHL	#164	4080	
		000000006	00 58	30	AE	D0 18	00402	578:	MOVL	SUBV	LIB\$SIGNAL VECTOR+8, R8	9658
					58	DD			MOVL BGEQ PUSHL PUSHL PUSHL CALLS	R8	VECTOR+8, R8	9660
		0000000G	00	00028EE8	8F 03	DD FB	0040C 00412		PUSHL	#167	7656 LIB\$SIGNAL #32	
			20		58	D1	00419 00410	58\$:	BLEQ	59\$		9662
		30	AE	00028073	20 8F 01	DO DD FB DO	0041E 00422 00428		CMPL BLEQ MOVL PUSHL CALLS	#163	SUBVECTOR+8	9665
		000000006	00 58	34	AE 16	D0 13	00426 0042F	59\$:	MOVL	SUBV	LIB\$SIGNAL VECTOR+12, R8	9669
			01		58	D1 13	0042F 00433 00435 00438		BEQL CMPL BEQL	R8,	#1	
					58	DD	0043A		BEQL PUSHL PUSHL PUSHL CALLS BISB2 BISB2 BISB2 BISB2 MOVW	# 1		9671
		000000006	00	00028140	8F 03	FB	00444		CALLS	#164	4160 LIB\$SIGNAL , a4(SP)	0474
		04	BE BE BE A6		02	88 88 80 80	0044B 0044F	60\$:	BISB2	#16,	a4(SP)	9676 9677 9678 9679
		10	A6	2C 30	04 AE AE 58 01	B0 B0	00457 00450		MOVW	SUBV	VECTOR+4, 16(R6) VECTOR+8, 18(R6)	9679
04 BE	01		03 A2 63		58	F0	00461	61\$:	MOVB	R8.	VECTOR+4, 16(R6) VECTOR+8, 18(R6) W3, W1, 24(SP) 31(NODEPTR)	9681 9682
				28	O1AT	D0	0046B 0046F		MOVL	20BA	VECTUR, (NUDESUBPIR)	9683 9500
20) A3	18 0A 10	A3	0E	50 50	9A DO	00472	62\$:	MOVZBL MOVL	STRI	IDE, 24(NODESUBPTR)	9700
20	,	ĭĉ	50 A3 A4 A2 51	18	50 A2	BO	00453 00457 00451 00467 00468 00472 00476 00478 00480 00484 00488		MOVAB	STRI	IDE, 28(NODÉPTR)	9681 9682 9683 9500 9699 9700 9703 9703
			01		50	D1 12	00/00		CMPL BNEQ	STRI 64\$	IDE, #1	
		02	06 A1	38	AE 06	E9	0048D 00491 00495 0049D 004AD 004AC 004AC 004BO		MOVL SUBL3 MOVW MOVAB CMPL BNEQ BLBC MOVB	SUBV	OSTPTR), STRIDE IDE, 24(NODESUBPTR) 10(DSTPTR), 32(NODESUBPTR) IDE, 28(NODEPTR) NODEPTR), R1 IDE, #1 VECTOR+16, 63\$ 2(R1) IDE, #2	9706 9708
		02	A1		02	90	00495	63\$:	BRB MOVB	#2.	2(R1)	9710 9706 9711
			02		50	01	0049D	64\$:		668		
		02	06 A1	38	AE 07	E9	004A2 004A6		BNEQ BLBC MOVB	SUBV	VECTOR+16, 65\$ 2(R1) 2(R1)	9713 9715
		02	A1		20 03 27	90	004AA	65\$:	BRB MOVB BRB	71\$	2(R1)	9717 9713

					M 5 16-Sep- 14-Sep-	-1984 02:10 -1984 12:17):13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 312 (37)
	04		50	D1 12	004B2 66\$:	CMPL BNEQ	STRIDE, #4	; 9718
02	06 A1	38	50 10 AE 08 18	12 E9 90	004B7 004BB	BLBC MOVB	68\$ SUBVECTOR+16, 67\$ #8, 2(R1) 71\$	9720 9722
02	A1			90	004BF 004C1 67\$:	BRB MOVB	71\$ 2(R1)	9724
02	06 A1	38	04 12 AE 01 04	E9 90	004C5 004C7 68\$: 004CB 004CF	BRB BLBC MOVB BRB	SUBVECTOR+16, 69\$ #1, 2(R1) 70\$	9724 9720 9727 9729
02 10	A1 A2 05		04 22 05 11 05	90 A4 D1 18 DD	004D1 69\$: 004D5 70\$: 004D9 71\$: 004DC 004DE	MOVB MULW2 CMPL BGEQ PUSHL	#34, 2(R1) #8, 28(NODEPTR) SUBSCR_COUNT, #5 72\$	9731 9732 9737
000000006	00 05	00028EA0	01 8F 03 55 11	DD DD FB D1 15 DD	004E0 004E2 004E8 004EF 72\$: 004F2	PUSHL PUSHL CALLS CMPL BLEQ PUSHL	#1 #167584 #3, LIB\$SIGNAL SUBSCR_COUNT, #5 73\$	9738
0000000G	00	00028EB0	01 8F 03 6E AE 06	DD DD FB E8	004F6 004F8 004FE 00505 73\$:	BLEQ PUSHL PUSHL PUSHL CALLS BLBS	#1 #167600 #3, LIB\$SIGNAL REF FLAG, 77\$ SUBVECTOR, RO	9743
	50	28	AE 06	D0 19	00508 0050C	MOVL	SUBVECTOR, RO	9746
06	A4		50	D1 19	0050E 00512	CMPL	RO. 6(DSTPTR)	9747
			50	DD	00514 745:	BLSS PUSHL PUSHL	RO	9750
		06	A4 02 8F	DD	00516 00519	PUSHL	6(DSTPTR) #2 #163963	9749
0000000G	00	0002807B	8F 04	DD DD DD DB	0051B 00521	PUSHI	#163963 #4. LIB\$SIGNAL	
	50	50	AE	DO 19	00521 00528 75\$:	MOVL	#4, LIB\$SIGNAL SUBVECTOR+4, RO 76\$	9752
0A	A4		50	D1	0052E	CMPL	RO. 10(DSTPTR)	9753
			50	19 DD	0052C 0052E 00532 00534 76\$:	BLSS CMPL BLSS PUSHL PUSHL PUSHL	RO 10(DSTPTR)	9756
		OA	50 A4 02 8F 04	DD	00536 00539 0053B	PUSHL	10(DSTPTR) #2	9755
0000000G	00	0002807B	8F	DD DD FB	0053B 00541	PUSHL	#163963	
FFFF8000	00 8F	30		D1 19	00548 77%:	CMPL	SUBVECTOR+8, #-32768	9759
00007FFF	8F	30	AE AE 12 AE 01	D1 15	00550 00552 0055A 0055C 78\$:	BLSS	SUBVECTOR+8, #32767	9760
		30	AE	00	0055A 0055C 78\$:	BLEQ PUSHL PUSHL PUSHL	79\$ SUBVECTOR+8	9762
		000280F0	01	DD DD	0055F 00561 00567	PUSHL		
000000006	00 58	34	8F 03 AE 11	FB 00 18	0056E 795:	MOVL BGEQ	#164080 #3, LIB\$SIGNAL SUBVECTOR+12, R8 80\$	9764
000000006	00	00028EE8	58 01 8F 03	DD DD FB	00572 00574 00576 00578 0057E	PUSHL PUSHL PUSHL CALLS	R8 #1 #167656 #3, LIB\$SIGNAL	9766

DBGPARSER V04-000			N 5 16-Sep- 14-Sep-	1984 02:10:13 VAX-11 Bliss-32 V4.0-742 1984 12:17:30 EDEBUG.SRCJDBGPARSER.B32;1	Page 313 (37)
		20	58 D1 00585 80\$: 11 15 00588	CMPL R8, #32 BLEQ 81\$; 9768
	34	AE		BLEQ 81\$	
	000000000	00028073	8F DD 0058F	MOVL #32, SUBVECTOR+12 PUSHL #163955	9771 9772
	00000000	5 00 58 38	AE DO 0059B 81\$:	MOVL SUBVECTOR+16, R8	9775
		01	16 13 0059F 58 D1 005A1	BEQL 82\$ CMPL R8. #1	
			11 13 005A4	BEQL 82\$	9777
		00000110	01 DD 005A8	MOVL #32. SUBVECTOR+12 PUSHL #163955 CALLS #1, LIB\$SIGNAL MOVL SUBVECTOR+16, R8 BEQL 82\$ CMPL R8, #1 BEQL 82\$ PUSHL R8 PUSHL #1 PUSHL #1 PUSHL #164160 CALLS #3, LIB\$SIGNAL	; 4111
	000000000	00028140	8F DD 005AA 03 FB 005B0	CALLS #3. LIB\$SIGNAL	
	1E	6 00 22 6B	6E E8 005B7 82\$: 13 E1 005BA	BLBS REF_FLAG, 84\$ BBC #19, (R11), 84\$ TSTL LOW_RANGE_VAL	9783 9785 9786
		00	5A D5 005BE	ISIL LUW_KANGE_VAL	9786
	06	A4	06 19 005C0 5A D1 005C2	CMDI I OLI DANGE VAI 6/DCTDTD)	9787
			5A D1 005C2 14 19 005C6 5A DD 005C8 83\$:	BLSS 84\$ PUSHL LOW RANGE_VAL PUSHL 6(DSTPTR) PUSHL #2 PUSHL #163963 CALLS #4, LIB\$SIGNAL BISB2 #16, @4(SP) BISB2 #2, @4(SP) BISB2 #4, @4(SP)	9790
		06	A4 DD 005CA 02 DD 005CD 8F DD 005CF	PUSHL LOW_RANGE_VAL PUSHL 6(DSTPTR)	9789
	00000000	0002807B	8F DD 005CF	PUSHL #2 PUSHL #163963	7109
	000000000	6 00 BE BE BE A6 30	04 FB 005D5 10 88 005DC 84\$:	CALLS #4, LIB\$SIGNAL BISB2 #16, a4(SP)	9795
	04 04 04 10	BE	02 88 005E0 04 88 005E4 AE BO 005E8 AE BO 005ED 58 FO 005F2 02 90 005F8	BISB2 #2, a4(SP) BISB2 #4, a4(SP)	9796
	10	A6 30 A6 34	AE BO 005E8 AE BO 005ED	MUVW SUBVELIUMED IDUMO)	9798
04 BE	01	03	AE BO 005E8 AE BO 005ED 58 FO 005F2	MOVW SUBVECTOR+12, 18(R6) INSV R8, #3, #1, 24(SP)	9800
	1F	A2	02 90 005F8 AE DO 005FC	MOVB #2, 31(NODEPTR)	9801
	25 14	A3 2C	AE DO 00600	MOVW SUBVECTOR+12, 18(R6) INSV R8, #3, #1, a4(SP) MOVB #2, 31(NODEPTR) MOVL SUBVECTOR, (NODESUBPTR) MOVL SUBVECTOR+4, 20(NODESUBPTR) BBC #19, (R11), 87\$ MOVL SUBVECTOR+4, 28(NODESUBPTR) MOVL SUBVECTOR+4, 32(NODESUBPTR)	9795 9796 9797 9798 9799 9800 9801 9802 9803
	2E 10	6B A3 2C A3 2C		BBC #19, (R11), 87\$ MOVL SUBVECTOR+4, 28(NODESUBPTR)	9811
	20	A3 2C	AE DO 00609 AE DO 0060E 13 E1 00613 85\$:	MOVL SUBVECTOR+4, 32(NODESUBPTR) BBC #19, (R11), 87\$	9811 9812 9830 9833
	28	6B AE	5A D1 00617	CMPL LOW_RANGE_VAL, SUBVECTOR	9833
		00028F08	0D 15 0061B 8F DD 0061D	MOVL SUBVECTOR+4, 28(NODESUBPTR) MOVL SUBVECTOR+4, 32(NODESUBPTR) BBC #19, (R11), 87\$ CMPL LOW_RANGE_VAL, SUBVECTOR BLEQ 86\$ PUSHL #167688 CALLS #1, LIB\$SIGNAL	
	000000000	63	01 FB 00623 5A DO 0062A 86\$:	MOVL LOW_RANGE_VAL, (NODESUBPTR)	9834
	08 00	A3 A3 28	5A DO 0062D	MOVL LOW_RANGE_VAL, 8(NODESUBPTR)	9835
	OC.	NJ 20	04 00636	MOVL SUBVECTOR, 12(NODESUBPTR)	9832
		24	A2 DD 00639	CLRL -(SP) PUSHL 36(NODEPTR)	9834 9835 9836 9832 9845 9845
		7E	02 DD 0063C	PUSHL 36(NODEPTR) PUSHL #2 MOVQ #6, -(SP)	9845
	2004		56 DD 00641	PUSHL R6	
	D2E6	CF	06 FB 00643 04 00648	CALLS #6, DBG\$BUILD_PRIMARY_SUBNODE	9848
. Danielas Cias	1400 5		***************************************		

; Routine Size: 1609 bytes, Routine Base: DBG\$CODE + 273E

DE

```
98523
98523
98553
985567
985557
985557
985557
985557
985557
985557
985557
985557
985557
985557
98557
98557
98557
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
985777
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
98577
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 9800
9801
9802
9803
9804
9805
9806
9807
9808
9809
```

VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1 ROUTINE GET_DEREFERENCE (PRIMPTR): NOVALUE = **FUNCTION** This routine is called upon seeing the dereference operator, e.g., the " in a PASCAL primary such as A". If the object being dereferenced is a pointer or a file variable, then this routine lights a bit in the current primary subnode which indicates that the dereference is taking place. It then calls DBG\$BUILD_PRIMARY_SUBNODE to append a new subnode. The type information in the new subnode reflects the type of the object being pointed to; or in the case of file variables, the type of the objects in the file. INPUTS PRIMPTR - A pointer to the Primary Descriptor currently being constructed by DBG\$PRIMARY_PARSER. OUTPUTS The Primary Descriptor pointed to by PRIMPTR is modified. BEGIN PRIMPTR: REF DBG\$PRIMARY; LOCAL FCODE. Local variable holding fcode info JUNK, Dummy output parameter NODEPTR: REF DBG\$PRIM NODE. Points to a Primary Sub-node TYPEID: Pointer to a RST type entry DBG\$GL_CURRENT_PRIMARY = .PRIMPTR; Check that the object being dereferenced is actually a pointer. IF .PRIMPTR[DBG\$B_DHDR_FCODE] NEQ RST\$K_TYPE_TPTR AND .PRIMPTR[DBG\$B_DHDR_FCODE] NEQ RST\$K_TYPE_FILE THEN SIGNAL (DBG\$_NOTPTR); Obtain a pointer to the bottom level sub-node by following the back-pointer. Light the EVAL bit in this subnode. which indicates that pointer dereferencing is taking place. Then, obtain the pointer to the RST type entry for the object being dereferenced. NODEPTR = .PRIMPTR [DBG\$L PRIM BLINK]; NODEPTR [DBG\$V PNODE EVAL] = TRUE;

TYPEID = .NODEPTR [DBG\$L_PNODE_TYPEID];

! from this typeid, get the typeid for the object being pointed to. ! for pointer variables, use the routine that extracts the typeid

			00	04 00000	GET_DERI	EFERENCE		
						.WORD SUBL2	Save R2	; 9849
	5E 52 00	04	08 522 132 08 102 08 102 08 102 103 104 105 105 105 105 105 105 105 105 105 105	C2 00002 D0 00005 D0 00009 91 00010		MOVL	Save R2 #8, SP PRIMPTR, R2	. 0001
000000006	00	04	52	00009		MOVI	R2. DRGSGI CURRENT PRIMARY	9881
	06	06	AZ I	91 00010		MOVL	R2. DBG\$GL_CURRENT_PRIMARY 6(R2), #6	; 9886
			13	13 00014		BEQL		
	OF	06	A2	91 00016 13 0001A		CMPB	6(R2), #15	; 9887
		000287F0	8F	13 0001A DD 0001C FB 00022 DO 00029 88 0002D DO 00031 91 00035		BEQL PUSHL	1\$ #165872	9889
0000000G	00		01	DD 0001C FB 00022 DO 00029		CALLS	#1, LIB\$SIGNAL	; ,,,,
	00 50 A0	18	A2 1	DO 00029	1\$:	MOVL BISB2	24(R2), NODEPTR	; 9899
0A	AU	00	01	88 0002b 00 00031 91 00035		BISB2	#1, LIB\$SIGNAL 24(R2), NODEPTR #1, 10(NODEPTR) 12(NODEPTR), TYPEID	9900
	6E 06	0C 06	AZ	91 00035		CMPR	6(R2) #6	9901
	••		OE	12 00039		MOVL CMPB BNEQ PUSHL	6(R2), #6 2\$:
			SE I	DD 0003B DD 0003D FB 00040		PUSHL	SP	; 9913
00000000	00	04	AE	DD 0003D		PUSHL	TYPEID	
00000000	UU		OF	11 00047		RRR	#2. DBG\$STA_TYP_TYPEDPTR	
			SE I	FB 00040 11 00047 DD 00049 9F 0004B	28:	CALLS BRB PUSHL	SP	9915
		08 08	SE AE AE O3	9F 0004B		PUSHAB	JUNK	
0000000G	00	08	AE	DD 0004E FB 00051 DD 00058 FB 0005A		PUSHL CALLS PUSHL	TYPEID #3, DBG\$STA_TYP_FILE	
00000000	00		6F	DD 00058	35:	PUSHI	TYPEID TYPEID	9916
0000000G	00		01	FB 0005A		CALLS	#1, DBG\$STA_TYPEFCODE	
			7E	D4 00061		CALLS CLRL PUSHL	-(SP)	; 9921
		04	AE	DD 00063 DD 00066		PUSHL	TYPEID	
	7E		06	DD 00063 DD 00066 7D 00068		PUSHL	#6, -(SP)	
			01 7E 8E 50 06 52	DD 0006B		MOVQ PUSHL	R2	
D273	CF			FB 0006D		CALLS	#6, DBG\$BUILD_PRIMARY_SUBNODE	
				04 00072		RET		: 9923

DE

DBGPARSER V04-000 0 6 16-Sep-1984 02:10:13 14-Sep-1984 12:17:30

VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32:1 Page 316 (38)

; Routine Size: 115 bytes, Routine Base: DBG\$CODE + 2D87

ROUTINE GET_FIELDREF (TOKEN): NOVALUE =

This routine picks up the position, size, and (optionally) the extension in a field reference (i.e., XX<pos,size,ext>. DBG\$EXPRESSION_PARSER is called to parse and evaluate each of these values. The values are stored as integers in the Operator Lexical Token entry.

This routine assumes that the opening angle bracket has already been found and that the parse pointer points to the start of the first expression in the field reference. When this routine returns, the parse pointer is left at the first character after the closing angle bracket.

TOKEN - a pointer to the Operator Lexical Token entry for the "<" operator.

OUTPUTS
The Lexical Token pointed to by TOKEN is modified to include the offset, size, and sign extension information.

BEGIN

MAP

TOKEN: REF TOKENSENTRY;

Pointer to the Lexical Token Entry for the field reference operator

DECLTYPE: REF DBG\$VALDESC,

SAVED_RADIX, VALUE,

VALPTR: REF DBG\$VALDESC:

Pointer to Value Descriptor
for one of the values
inside the angle brackets
Temporarily saved expression radix
Value of position, size, or sign ext
field
Pointer to position, size, or
extension value descriptor.

Loop through the expressions in this field reference. Each of these is parsed and evaluated via a call to DBG\$EXPRESSION_PARSER. This returns a descriptor, and the type converter is then called to convert the descriptor into an integer value. The integer value is checked for being in an appropriate range and then stored in the appropriate own variable.

INCR I FROM 1 TO 3 DO

Call the expression parser to pick up the next expression in the field reference. Note that we set the radix to decimal over this call and then restore it. Also note that the Expression Parser sets TERMINATOR_CODE and TERMINATOR_LENGTH as a side-effect.

! must be either 0 or 1. We also insist that the terminator must

Page 318 (39)

```
6
DBGPARSER
V04-000
                                                                                                16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                     VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32;1
  9943
9944
9946
9946
9948
9949
9951
9955
9956
9966
9966
9966
9966
                                                               be the closing angle bracket in this case.
                                                            IF .I EQL 3
                                                                  BEGIN
                                                                      . VALUE NEQ O AND . VALUE NEQ 1
                                                                  THEN
                                                                        SIGNAL (DBGS_ILLSIGEXT, 1, .VALUE);
                                                                  TOKEN [TOKEN$V_SGNEXT] = .VALUE;
IF .TERMINATOR_CODE NEG TOKEN$K_TERM_GTRTHAN
THEN_____
                                                                        SIGNAL (DBG$_INVFLDREF);
                                                                  EXITLOOP:
                                                                  END:
                                                            END:
                                                                                                            ! End of INCR loop
                        10058
                                                         All done. Return to the caller.
                        10060
                                                      RETURN;
                        10061
                                                      END:
                                                                                   OOFC 00000 GET_FIELDREF:
                                                                                                                           Save R2,R3,R4,R5,R6,R7
LIB$SIGNAL, R7
                                                                                                                                                                                                 9924
                                                                                                                . WORD
                                                                                     9E
9E
00
                                                               00000000g
                                                                                          00002
                                                                                                               MOVAB
                                                                                                               MOVAB
                                                                                                                           TERMINATOR_CODE, R6
                                                                                          00010
00013
00017
                                                                                                                                                                                                 9973
9982
9983
                                                                                01
                                                                                                               MOVL
                                                                                                                           EXPRÉSSION RADIX, SAVED RADIX
#10, EXPRESSION RADIX
BIT SELECT TERM TBL
                                                                                     000F4B00D52DB001259
                                                                        08
                                                                               A0EFE2005669F16551030292D21
                                                                                                               MOVL
                                                   08
                                                                                                               MOVL
                                                                                          0001B
00021
00023
0002B
0002F
00031
00039
00042
00044
00049
                                                               00000000
                                                                                                               PUSHAB
                                                                                                                                                                                                 9984
                                                                                                                           -(SP)
                                                                                                               CLRL
                                                                                                                           #2. DBG$EXPRESSION_PARSER
                                                D7D6
                                                                                                               CALLS
                                                                                                               MOVL
                                                                                                                           RO. VALPTR
                                                                                                                           SAVED_RADIX, EXPRESSION_RADIX
                                                                                                                                                                                                 9985
9992
                                                   08
                                                           A6
                                                                                                               MOVL
                                                                                                                           TERMINATOR_CODE
                                                                                                               TSTL
                                                                                                               BNEQ
                                                               00028E90
                                                                                                               PUSHL
                                                                                                                            #167568
                                                                                                                           #1, LIB$SIGNAL
TERMINATOR_LENGTH, CHARPTR
                                                                                                               CALLS
ADDL2
                                                                                                                                                                                                9993
9998
                                                FBCC
                                                           6
                                                                        04
                                                                                                   25:
                                                                                                               PUSHL
                                                                                                                           VALPTR
                                                          CF
52
01
                                                                                                                CALLS
                                                                                                                           #1. CONVERT_TO_INTEGER
                                                F65A
                                                                                                                           RO. VALUE
                                                                                                               MOVL
                                                                                                                                                                                                 1000
                                                                                                               BNEQ
                                                                                                               TSTL
                                                                                                                            VALUE
                                                                                                                                                                                                 1000
                                                                                                               BLSS
                                                                                     D1
                                          00007FFF
                                                          8F
                                                                                                               CMPL
                                                                                                                            VALUE, #32767
                                                                                                               BLEQ
                                                                                                   38:
                                                                                                               PUSHL
                                                                                                                                                                                                 1001
                                                                                                                           VALUE
                                                                                                               PUSHL
```

DBGPARSER V04-000					H 6 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 320 (39)
		08	00028EC8 67 50 04 A0 0A	8F 03 AC 52 66	DD 00062 FB 00068 DO 0006B DO 0006F DO 0006F DO 0006F DO 0006F DO 00076 DO 00076 DD 00076 DD 00078 FB 00076 DD 00078 FB 00078 FB 00078 CALLS DD 00081 CALLS M1, LIBSSIGNAL DO 00084 DS CALLS DS 00086 TSTL VALUE DS 00086 TSTL VALUE DS 00087 DO 00097 DO 00097 DO 00097 DO 00097 DO 00097 DO 00097 DO 00098 DS CALLS M1, LIBSSIGNAL DS 00099 CALLS M3, LIBSSIGNAL CMPL CMPL CMPL CMPL CMPL CMPL CMPL CMP	1001
			67 00028F00	66 09 8F 01	12 00076 BNEQ 55 DD 00078 PUSHL #167680 FB 0007E CALLS #1, LIB\$SIGNAL	1001
			67 02	53	FB 0007E CALLS #1, LIB\$SIGNAL D1 00081 5\$: CMPL 1, #2 12 00084 BNEQ 8\$	1002
			20	50	D1 00081 5\$: CMPL I, #2 12 00084 BNEQ 8\$ D5 00086 TSTL VALUE 19 00088 BLSS 6\$ D1 0008A CMPL VALUE, #32 15 0008D BLEQ 7\$ DD 0008F 6\$: PUSHL VALUE DD 00091 PUSHL #1 DD 00093 PUSHL #167632 FB 00099 CALLS #3, LIB\$SIGNAL	1002
			00028ED0	01 8F 03	DD 0008F 6\$: PUSHL VALUE DD 00091 PUSHL #1 DD 00093 PUSHL #167632	1002
		0A	50 04	03 AC 52	DD 00093 PUSHL #167632 FB 00099 CALLS #3, LIB\$SIGNAL DO 0009C 7\$: MOVL TOKEN, RO BO 000A0 MOVW VALUE, 10(RO) D1 000A4 CMPL TERMINATOR_CODE, #10	1003
			0A	66	D1 000A4 CMPL TERMINATOR_CODE, #10 13 000A7 BEQL 11\$	1003
			03	53	D1 000A9 88: CMPL I #3 12 000AC BNEQ 108	1004
			01	66 353 52 52 52 52 52 52 52 52 52 52 52 52 52	DO 0009C 7\$: MOVL TOKEN, RO BO 000A0 MOVW VALUE, 10(RO) D1 000A4 CMPL TERMINATOR_CODE, #10 13 000A7 BEQL 11\$ 12 000AC BNEQ 10\$ D5 000AE TSTL VALUE 13 000B0 BEQL 9\$ D1 000B2 CMPL VALUE, #1 13 000B5 BEQL 9\$ DD 000B7 PUSHL W1 DD 000B9 PUSHL #1 DD 000BB FB 000C1 FO 000C4 9\$: INSV VALUE, #10, #1, atoken D1 000CA CMPL TERMINATOR_CODE, #10 13 000CD BEQL 11\$ DD 000CF PUSHL #167680 FB 000D5 CALLS #1, LIB\$SIGNAL FB 000D6 CALLS #1, LIB\$SIGNAL FB 000D7 PUSHL #167680 FB 000D7 CALLS #1, LIB\$SIGNAL FB 000D7 TST. ACBL #3, #1, I, I\$ O4 000D8 TST. ACBL #3, #1, I, I\$	1004
			00028140	8F	DD 000B7 PUSHL VALUE DD 000B9 PUSHL #1 DD 000BB PUSHL #164160	1004
04 BC	01		67 0A 0A	03 52 66 10 8F 01	DD 000BB PUSHL #164160 FB 000C1 CALLS #3, LIB\$SIGNAL FO 000C4 9\$: INSV VALUE, #10, #1, atoken D1 000CA CMPL TERMINATOR_CODE, #10 13 000CD BEQL 11\$ DD 000CF PUSHL #167680 FB 000D5 CALLS #1, LIB\$SIGNAL	1004 1004
			67 00028F00	8F	DD 000CF PUSHL #167680	1005
****	.,				FB 000D5 CALLS #1, LIB\$SIGNAL 04 000D8 RET	1004
FF34	53		01	03	F1 000D9 10\$: ACBL #3, #1, I, 1\$ 04 000DF 11\$: RET	1004 9973 1006

; Routine Size: 224 bytes, Routine Base: DBG\$CODE + 2DFA

```
9968
9970
9971
9973
9974
9975
9976
9977
9978
9981
9981
9983
9984
9983
9984
9986
9991
9991
9993
9993
9994
9996
9997
9997
9998
10001
10012
10013
10016
10016
10016
10017
10018
10018
10018
10018
10018
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
10019
1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         10062
10063
10064
10065
10066
10067
10070
10071
10073
10074
10075
10076
10079
10080
10081
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  10082
10083
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  10084
10085
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              10086
10087
10088
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          10089
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             10089
10090
10091
10092
10093
10094
10095
10096
10097
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  10100
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              10101
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     10102
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 10104
10105
10106
10107
10108
10109
10110
10111
10112
10113
10114
10115
10116
10117
```

ROUTINE GET_RECORD_COMPONENT(PRIMPTR, COMPNAME): NOVALUE = FUNCTION

This routine is called during the parsing of Primary Symbols to do record component selection. It accepts as input a Primary Descriptor for a record and the name of a record component to be selected from that record. It then checks that the Primary Descriptor is indeed for a record (otherwise component selection is not allowed and an error is signalled). It then looks up the component name in the RST and gets the SYMID for the specified component of the specified record. If no such component exists for this record, an error is signalled. Finally, this component SYMID is converted to a record component index which is stored in the Record Sub-Node in the Primary Descriptor and another Sub-Node is appended for the record component itself. The output of the routine is thus the side-effect of modifying the input Primary Descriptor.

INPUTS

PRIMPTR - A pointer to the Primary Descriptor for the record on which component selection is to be done.

COMPNAME - A pointer to the name of the record component to be selected. The name must be in Counted ASCII format.

OUTPUTS

The PRIMPTR Primary Descriptor is modified by filling in the record component index for the selected component and by appending another Primary Descriptor Sub-Node for the component. The PRIMPIR pointer itself is not modified, however.

BEGIN

PRIMPTR: REF DBG\$PRIMARY;

COMP_LIST: REF VECTOR[],

COMPSYMID: REF RSTSENTRY, EXACT_MATCH, FCODE, NODEPTR: REF DBG\$PRIM_NODE,

STATUS, SYMID: REF RSTSENTRY, TYPCOMPLST: REF VECTOR[,LONG],

TYPEID: REF RSTSENTRY,

VARPTR: REF RST\$VAR_ENTRY, VARSETPTR: REF RSTSENTRY:

DBG\$GL_CURRENT_PRIMARY = .PRIMPTR;

! Pointer to Primary Descriptor

List of potential component symids Length of COMP_LIST Length of COMP_LIST
SYMID for current record component
Flag saying we've found the record component
The FCODE of the record component
Pointer to Record Sub-Node in the
Primary Descriptor
Status returned by GET_RECORD_VARIANT
The SYMID of the record component
Pointer to list of record component
SYMIDs in record Type RST Entry
The Type ID of the record record type
or of the record component or of the record component
Pointer to current RST Variant Entry
Pointer to Variant Set RST Entry

for languages which allow non-records to be component-selected (i.e., language (), we do not do the above check. In fact, we explicitly change the FCODE to say 'record' so that the Primary is processed as a record by routines in DBGVALUES and in DBGPRINT. ELSE

Page 322 (40)

NODEPTR[DBG\$B_PNODE_FCODE] = RST\$K_TYPE_RECORD;

10166 10167

10168 10169 10170

:10081

! Search the RST Hash Table for all record components of this name. ! If we find one which belongs to the given record, then light the

IF NOT .EXACT_MATCH

Page 323 (40)

```
DBGPARSER
V04-000
                                                                                                                       16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                   VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
:10196
:10197
:10198
                                                                                  EXITLOOP;
END;
                                                                              If this record component is a Variant Set, see if the desired component is part of one of the variants in this Variant Set.
                                                                           IF .COMPSYMID[RST$B_KIND] EQL RST$K_VARIANT
                                                                           THEN
10205
10206
10207
10208
10209
10210
10211
10213
10213
10215
10216
10217
10221
10221
10223
10223
10226
10227
10228
                                                                                  BEGIN
                                                                                  VARSTK_INDEX = 0;
STATUS = GET_RECORD_VARIANT(.COMPSYMID, .SYMID);
                                                                                  IF .STATUS
                                                                                         BEGIN
                                                                                             We found the record component in the current Variant Set. Set the index of the Variant Set in the Record Sub-Node.
                                                                                             Then build all necessary Primary Descriptor Variant Sub-
Nodes, one for each level of variant nesting.
                                                                                         NODEPTREDBG$W_PNREC_INDEX] = .I + 1;
INCR_J_FROM_O_TO_.VARSTK_INDEX - 1 DO
                                                                                                 BEGIN
                                                                                                DBG$BUILD_PRIMARY_SUBNODE(.PRIMPTR, RST$K_VARIANT, 0, 0);
                                                                                                VARSETPTR = .VARSTACK1[.J];
VARPTR = .VARSTACK2[.J];
                                                                                                NODEPTR[DBG$V_PNODE_EVAL] = TRUE;
NODEPTR[DBG$V_PNODE_EVAL] = TRUE;
NODEPTR[DBG$V_PNVAR_TAGID] = .VARSETPTR[RST$L_VARTAGPTR];
NODEPTR[DBG$W_PNVAR_INDEX] = .VARSTACK3[.J];
NODEPTR[DBG$W_PNVAR_NCOMPS] = .VARPTR[RST$L_VAR_COMPCNT];
NODEPTR[DBG$L_PNVAR_COMPLST] = VARPTR[RST$A_VAR_COMPLST];
NODEPTR[DBG$L_PNVAR_DSTPTR] = .VARPTR[RST$L_VAR_DSTPTR];
                                                                                                END:
                                                                                             The Variant Sub-Nodes have successfully been constructed. Now exit the search of the record component list so that
                                                                                             we can build the Sub-Node for the component actually found.
                                                                                         EXITLOOP:
                                                                                         END:
                                                                                  END:
                                                                                                                                     ! End of variant code
                                                                          END:
                                                                                                                                     ! End of loop over record components
                                                                      finally append another Primary Descriptor Sub-Node for the selected
                                                                      record component. Then return.
                                                                   DBG$STA_SYMTYPE(.SYMID, FCODE, TYPEID);
DBG$BUIED_PRIMARY_SUBNODE(.PRIMPTR,
10251
                                                                                                                       RST$K_TYPCOMP, .SYMID, .FCODE, .TYPEID, 0);
```

Page 325 (40)

	PARS -000 253 254 255	ER		103 103 103	47 48 49	221			RE1	TURN,	•				1	N 6 6-Sep-19 4-Sep-19	284 02:10 84 12:17	:13 VAX-11 Bliss-32 V4.0-742 P :30 [DEBUG.SRC]DBGPARSER.B32;1	age 326 (40)
SF 4E	54	45 4E	47 4F	5C 50	52 40	45 4F	53 43	52 5F	41	50 52	47 4F 30	4243	44 45 20	03 44 42 55 4	03372 03373 03374 03375 03376 03385 03394	P.AYJ: P.AYK:	.PSECT .BYTE .ASCII .ASCII .ASCII	DBG\$PLIT,NOWRT, SHR, PIC,0 3 \A\ \L\ \L\ \!DBGPARSER\<92>\GET_RECORD_COMPONENT 10\	
							000	00000 FE7 FE6	00° 78 6A	09 EF CF 06 CF 57 07	0002	0000	AAF05CB8A1 A9A1A01FAAA2C1FAAAA2C1FAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAA2C1FAAAAA2C1FAAAA2C1FAAAA2C1FAAAAAAAAAA	OFF C 200000912000000000000000000000000000000	00010 00010 00018 00024 00024 00035 00035 00035 00035 00045 00045 00045 00065 00065	1\$: 2\$: 3\$:	PSECT COMP COMP SUBLE MOVL MOVL MOVL MOVL MOVL MOVL MOVL MOVL	24(R10), NODEPTR ENFORCE_RECORD, 5\$ DBG\$GB_CANGUAGE, #9 3\$ aCOMPNAME, P.AYJ 1\$ R10 #1, GET_DEREFERENCE 6(R10), #6 2\$ R10 #1, GET_DEREFERENCE 1\$ 24(R10), NODEPTR 6(R10), #7 4\$ COMPNAME #1 #167576 #3, LIB\$SIGNAL 9(NODEPTR), #7 6\$ P.AYK #1 #164706	1006 1011 1012 1013 1013 1014 1014 1014 1015 1015 1015 1015
							000	00000	00G 09 08	00 A7 AE 56		00	03 04 07 A7 6E 0A	90 90	0007C 0007E 00082 00087	5\$: 6\$:	CALLS BRB MOVB MOVL CLRQ MOVL	#3, LIB\$SIGNAL 6\$ #7, 9(NODEPTR) 12(NODEPTR), TYPEID EXACT_MATCH #10, COMP_LIST_SIZE	1012 1017 1018 1018 1018

					1	B 7 6-Sep- 4-Sep-	1984 02:10 1984 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 327 (40)
	0000000G	00		50109CB1B10ED887851B99615A010	DD 00086 FB 00086 D0 00096 D0 00096 FB 000A6 DD 00086 DD	5	PUSHL CALLS MOVL	COMP_LIST_SIZE #1, DBG\$GET_TEMPMEM R0, COMP_LIST (COMP_LIST) COMPNAME, R11	1018
		5B	08	AC	D4 00098		MOVL CLRL MOVL PUSHL CALLS PUSHL CALLS	(COMP LIST) COMPNAME, R11	: 1018 : 1018
	0000000G	00		01	DD 00096 FB 000A0 DD 000A7)	CALLS	#1 DBG\$HASH_FIND_SETUP	1010
	00000000G	00		01	FB 000A9	7\$:	CALLS	#1, DBG\$HASH_FIND RO, SYMID SYMID, R8	1019
		AE 58	04	AE 3D	FB 000A9 00 000B0 00 000B4 13 000B8		MOVL MOVL BEQL CMPB	SYMID, R8	1019
		OA	14	A8 E7	91 000B/ 12 000B		CMPB BNEQ	20(R8), #10	1019
	08	AE	10	A8 05	D1 000C0		CMPL BNEQ	16(R8), TYPEID	1019
		6E		01 2B	01 00000 12 00000 00 00000 11 00000		MOVI	#1, EXACT_MATCH	1020
		56		69	D6 000CC	HE.	BRB INCL CMPL	(COMP_LIST) (COMP_LIST), COMP_LIST_SIZE	; 1020 ; 1021
		52		1B 59	01 00006 19 00003 9F 00006 FB 00006		BLSS MOVL PUSHAB		1021 1021
	0000000G	00	0A	01 01	FB 00005		CALLS	#1, DBG\$GET_TEMPMEM	; 1021
50		59		95	FB 00005 78 000E3 28 000E3 CO 000EE DO 000EE DO 000F3)	CALLS MOVL ASHL MOVC3 ADDL2	COMP_LIST, SAVE_COMP_LIST 10(COMP_LIST_SIZE) #1, DBG\$GET_TEMPMEM R0, COMP_LIST #2, COMP_LIST SIZE, R0 R0, (SAVE_COMP_LIST), (COMP_LIST) #10, COMP_LIST_SIZE (COMP_LIST), R0 R8, (COMP_LIST)[R0]	1022
07		56 56 56 50 940		02 05 06 58 86 6F EF	CO 000EE		ADDL2	#10, COMP_LIST_SIZE	1022
	(940		58	DO 000EE DO 000F1 11 000F5		MOVL	R8. (COMP_LIST)[RO]	
		42	00000000	6E	E8 000F7	105:	MOVL MOVL BRB BLBS BLBC BLBC	EXACT MATCH, 15\$; 101 <u>8</u> ; 1023 ; 1023
		04	0000000	-	E9 00101		1511	ENFORCE RECORD, 11\$ INCOMPLETE QUAL, 12\$ (COMP LIST)	1024
				0C 5B	12 0010/ DD 00100	12\$:	BNEQ	(COMP_LIST) 13\$ R11	1025
			00028080	01 8F	D5 00108 12 00100 DD 00108 DD 00110		PUSHL	#1 #167040	
		7E		69 058 058 108 108 108 108 108 108 108 108 108 10	7D 00118	11\$: 12\$: 13\$: 14\$: 15\$:	BNEQ PUSHL PUSHL PUSHL BRB MOVQ PUSHAB	14\$ R10, -(SP) SYMID	1026 1025
			00	AE 59	9F 0011E		PUSHAB	SYMID COMP_LIST	: 1025
	0000v	CF 11	18	AE 05	FB 00123		PUSHL PUSHL CALLS BLBS PUSHL PUSHL PUSHL CALLS MOVL BISB2	COMP_LIST TYPEID #5, RESOLVE_COMPONENT	
		11			DD 00128		PUSHL	RO, 15\$ R11 #1	1026
	00000000	00	00028F58	8F	DD 00126		PUSHL	#167768	
	000000006	57	18	AA	00 0013	158:	MOVL	24(R10), NODEPTR	1026
	0A	58	08 20	AE	00 00144		WOAL	#167768 #3, LIB\$SIGNAL 24(R10), NODEPTR #1, 10(NODEPTR) TYPEID, R8 44(R8), TYPCOMPLST	1026 1026 1027
		58 52 54	20	5B 01 8F 03 AA 01 AE A8 01 0086 6244	7D 00118 9F 00118 DD 00128 E8 00128 DD 00128 DD 00128 DD 00128 DD 00138 B8 00148 CE 00148 31 00148		MOVL MOVAB MNEGL	#1, I 20\$	1028
		56		6244	00 00152	16\$:	BRW MOVL	(TYPCOMPLST)[1], COMPSYMID	1028

DB VO

DBGPARSER V04-000		C 7 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 328 (40)
18 A7	56 04 AE 07 54 01 7F 0B 14 A6 000000000 EF	D1 00156	; 1028 : 1028 : 1028 : 1029 : 1030 : 1030
18 A7	00000000° EF 04 AE 56 02 50 6E 59 54 58 00000000° EF 53	DO 00179 MOVL RO, STATUS E9 0017C BLBC STATUS, 20\$ A1 0017F ADDW3 #1, I, 24(NODEPTR) DO 00184 MOVL VARSTK_INDEX, R11 CE 0018B MNEGL #1, J	1030 1030 1031 1031
CFF	59 00000000°EF43 55 00000000°EF43 57 18 AA	70 00190 185: CLRQ -(SP) DD 00192 PUSHL #19 70 00194 MOVQ #11, -(SP)	1031 1031 1031 1031
BA 02		F7 001BB	1031 1031 1032 1032 1032 1032 1032 1032
0000000	08 AE 10 AE 0C AE	31 001DF 21\$: BRW 16\$ 9F 001E2 22\$: PUSHAB TYPEID 9F 001E5 PUSHAB FCODE DD 001E8 PUSHL SYMID FB 001EB CALLS #3, DBG\$STA_SYMTYPE D4 001F2 CLRL -(\$P) DD 001F4 PUSHL TYPEID	1034 1034 1034
CF8	10 AE 0A 5A 06	DD 001F7	1034

; Routine Size: 519 bytes, Routine Base: DBG\$CODE + 2EDA

V(

ROUTINE GET_RECORD_VARIANT(VARSETPTR, SYMID) =

FUNCTION This routine looks for a record component with a known SYMID among all the variants in a specified Variant Set. It returns TRUE if the component is found in that Variant Set, and as a side-effect, it also builds a "Variant Stack" which specifies which sequence of variants and Variant Sets contain the component. This routine calls itself recursively to search variants within variants; the "Variant Stack" records the path taken through the tree of variants to reach the desired component. This stack is then used by GET_RECORD_COMPONENT to build the Primary Descriptor Variant Sub-Nodes needed to describe that path.

INPUTS VARSETPTR - Pointer to the Variant Set to be searched for the SYMID record component.

SYMID - The SYMID (RST Entry address) of the record component to search for among the variants of the current Variant Set. (This SYMID has been found by looking up the record component by its name.)

OUTPUTS If the SYMID record component is found in any of the VARSETPTR variants, this routine returns TRUE as its value. If the SYMID component is not found, it returns FALSE. If TRUE is returned, the Variant Stack (in OWN storage) contains the list of Variance Component is not found. ant Sets and variants which contain the SYMID component.

BEGIN

VARSETPTR: REF RSTSENTRY;

! Pointer to Variant Set RST Entry

COMPLST: REF VECTOR[,LONG],

COMPPTR: REF RSTSENTRY.

VARPTR: REF RST\$VAR ENTRY. VARSETTBL: REF VECTOR[,LONG],

STATUS:

Pointer to vector of component RST pointers in RST Variant Entry Pointer to current variant component's RST entry Pointer to current RST Variant Entry Pointer to vector of variant in the Variant Set RST Entry Status returned by recursive call

Push the address of the current Variant Set RST Entry on the Variant Stack maintained by this routine and GET_RECORD_COMPONENT. This is how we keep track of nested Variant Sets in the record.

if .varstk_index geq varstk_size then signal(dbgs_varnesdep);
varstk_index = .varstk_index + 1;
varstack1[.varstk_index - 1] = .varsetptr;

! Loop through all the variants in this Variant Set. For each variant in

```
DBGPARSER
V04-000
                                                                                                                            16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                          VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGPARSER.B32;1
                                                                                                                                                                                                                                               Page 330
(41)
                                                                         the set, search its components until we find the SYMID component, i.e
                                                                         the record component we are looking for.
                                                                      VARSETTBL = VARSETPTR[RST$A_VARSETTBL];
INCR I FROM 0 TO .VARSETPTR[RST$L_VARSETCNT] - 1 DO
                                                                             BEGIN
                                                                             VARPTR = .VARSETTBL[.1];
VARSTACK2[.VARSTK_INDEX - 1] = .VARPTR;
                                                                                Loop over the record components in this particular variant. If the SYMID component is found, we return TRUE and leave the Variant Stack with its current contents. If we find another Variant Set component within this variant (i.e., nested variants within the record), this routine calls itself recursively to look for the SYMID component in that nested variant. If it is found there, TRUE is returned.
                                                                             COMPLST = VARPTR[RST$A VAR COMPLST]:
INCR J FROM 0 TO .VARPTR[RST$L_VAR_COMPCNT] - 1 DO
                                                                                     BEGIN
                                                                                    COMPPTR = .COMPLST[.J];

VARSTACK3[.VARSTK_INDEX - 1] = .J + 1;

IF .COMPPTR EQL .SYMID THEN RETURN TRUE;

IF .COMPPTR[RST$B_KIND] EQL RST$K_VARIANT
                                                                                     THEN
10339
10340
10341
10342
10343
10344
10346
10347
10350
10351
10352
                                                                                             BEGIN
                                                                                            STATUS = GET_RECORD_VARIANT(.COMPPTR, .SYMID);
IF .STATUS THEN RETURN TRUE;
                                                                                             END:
                                                                                     END:
                                                                                                                                           ! End of loop over variant components
                                                                             END:
                                                                                                                                           ! End of loop over Variant Set
                                                                         We did not find the SYMID component anywhere among the variants in this
                                                                         Variant Set. We thus pop the Variant Stack and return FALSE.
                                                                     VARSTK_INDEX = .VARSTK_INDEX - 1;
RETURN FALSE;
:10354
:10355
                                                                     END:
                                                                                                          OSFC 00000 GET_RECORD_VARIANT:
                                                                                                                                                             Save R2,R3,R4,R5,R6,R7,R8,R9
VARSTK_INDEX, R9
VARSTK_INDEX, #20
                                                                                                                                                                                                                                                      1035
                                                                                 00000000
                                                                                                                                              MOVAB
                                                                                                      EF 69 00 8F 01 69 69 AC
                                                                                                                   00002
00000
0000E
00014
0001B
0001D
                                                                                                              D1
19
                                                                                                                                                                                                                                                      1040
                                                                                                                                               CMPL
                                                                                                                                              BLSS
                                                                                                              DD 860000
                                                                                 00028A32
                                                                                                                                               PUSHL
                                                                                                                                                              #166450
                                                                                                                                                              #1, LIB$SIGNAL
VARSTK_INDEX
VARSTK_INDEX, RO
VARSETPTR, R7
                                                      0000000G
                                                                                                                                               CALLS
                                                                                                                                                                                                                                                      1040
                                                                                                                               15:
                                                                           50
                                                                                                                                               MOVL
```

MOVL

04

DBGPARSER V04-000				F 7 16-Sep-1984 02:10:13 VAX-1 14-Sep-1984 12:17:30 EDEBU	1 Bliss-32 v4.0-742 Page 331 G.SRCJDBGPARSER.B32:1 (41)
		FFOC C940 53 56	18	0 00024 MOVL R7, VARSTAC 0 0002A MOVAB 24(R7), VAR 0 0002E MNEGL #1, I 1 00031 BRB 6\$	K1-4[R0] SETTBL : 1041 : 1042
		52 51 FF5C C941	63	0 00031 0 00033 2\$: MOVL (VARSETTBL) 0 00037 MOVL VARSTK_INDE 0 0003A MOVL VARPTR, VAR	[]], VARPTR 1041 X, R1 1041 STACK2-4[R1] 1042
		54	08	E 00040 MOVAB 8(R2), COMP E 00044 MNEGL #1, J 1 00047 BRB 5\$	1042
		58 51 AC A941 08 AC	01	0 00049 3\$: MOVL (COMPLST)[J 0 00040 MOVL VARSTK_INDE 0 00050 MOVAB 1(R5), VARS 1 00056 CMPL COMPPTR, SY	J. COMPPTR X. R1 TÁCK3-4[R1] MID : 1042
		08	14 08	3 0005A BEQL 48 1 0005C CMPB 20(COMPPTR) 2 00060 BNEQ 58	
		95 AF 04 50	08	00 00033 2\$: MOVL (VARSETTBL) 00 00037 MOVL VARSTK_INDE 00 00038 MOVL VARPTR, VAR 00 00040 MOVAB 8(R2), COMP 10 00047 BRB 5\$ 10 00047 BRB 5\$ 10 00040 MOVL VARSTK_INDE 10 00040 MOVL VARSTK_INDE 10 00050 MOVAB 1(R5), VARS 11 00056 CMPL COMPPTR, SY 10 00050 BEQL 4\$ 10 00050 BEQL 4\$ 10 00050 BRBQ 5\$ 10 00050 BRBQ 5\$ 10 00065 BRBQ 5\$ 10 00065 BNBQ 5	
	D2 B7	55 56	04 08	A 00071 2 00072 5\$: AOBLSS 4(VARPTR), 2 00077 6\$: AOBLSS 8(R7), I, 2 0007C DECL VARSTK_INDE 4 0007E CLRL RO RET	J. 3\$ 1042 \$ 1041 X 1044 1044

; Routine Size: 129 bytes, Routine Base: DBG\$CODE + 30E1

END:

VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32:1

```
Loop through the set constant expressions for this set constant. Each set constant is parsed, evaluated, and converted to the appropriate type of the first set constant (with the type being checked in the process). Note that TERMINATOR CODE is set within the loop as a side-effect of the call on DBG$EXPRESSION_PARSER.
```

CREATE = TRUE;
THIS_SUBSCR_IS_RANGE = FALSE;
TERMINATOR_CODE = TOKENSK_TERM_COMMA;
WHILE .TERMINATOR_CODE NEW TOKENSK_TERM_CLOSE DO
BEGIN

Call the expression parser to pick up the next set constant expression and its value. Note that we set the radix to decimal over this call and then restore it. Also note that the Expression Parser sets TERMINATOR_CODE and TERMINATOR_LENGTH as a side-effect.

SAVED_RADIX = .EXPRESSION_RADIX; EXPRESSION_RADIX = DBG\$K_DECIMAL; VALPTR = DBG\$EXPRESSION_PARSER (FALSE, SET_CONSTANT_TERM_TBL); EXPRESSION_RADIX = .SAVED_RADIX;

Check the terminator code. If there was no terminator (the input line just ended), signal an error. Otherwise we got a comma or closing subscript parenthesis and we increment CHARPTR to get past it.

IF .TERMINATOR_CODE EQL TOKENSK_TERM_NONE THEN SIGNAL(DBGS_MISCLOSUB); CHARPTR + .TERMINATOR_LENGTH;

Create a set constant value descriptor. Its type is the type of the first set constant data type.

```
IF .CREATE
THEN

BEGIN

SETVALPTR = DBG$MAKE_SKELETON_DESC(DBG$K_VALUE_DESC, 8*4);

SETVALPTR[DBG$B_DHDR_LANG] = .DBG$GB_LANGUAGE;

SETVALPTR[DBG$B_DHDR_KIND] = RST$K_DATA;

SETVALPTR[DBG$B_DHDR_FCODE] = RST$K_TYPE_SET;

SETVALPTR[DBG$W_VALUE_LENGTH] = 32;

SETVALPTR[DBG$L_VALUE_POINTER] = SETVALPTR[DBG$A_VALUE_ADDRESS];

SETVAL = .SETVALPTR[DBG$L_VALUE_POINTER];

SELECTONE .VALPTR[DBG$B_DHDR_FCODE] OF

SET

[RST$K_TYPE_ATOMIC, RST$K_TYPE_DESCR]:

BEGIN

SETVALPTR[DBG$L_DHDR_TYPEID] = DBG$TYPEID_FOR_SET(

.VALPTR[DBG$B_VALUE_DTYPE], RST$K_TYPE_SET, 256, TRUE);

END;
```

[RST\$K_TYPE_ENUM]:

```
DBGPARSER
V04-000
                                                                                                         VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32;1
                                                       Otherwise, set the low range value to be the set constant value.
                                                    ELSE
                                                          LOW_RANGE_VAL = .VALADDR[0];
                                                     END:
                                                  Set the set value.
                                                IF .LOW_RANGE_VAL LSS 0 OR .LOW_RANGE_VAL GTR 256 THEN
                                                     SIGNAL (DBG$_BITRANGE);
                                                IF .VALADDR[0] LSS 0 OR .VALADDR[0] GTR 256 THEN
                                                     SIGNAL (DBG$_BITRANGE);
                                                INCR I FROM .LOW_RANGE_VAL TO .VALADDR[0] DO SETVAL[.1] = TRUE;
                                                                                      ! End of WHILE loop over set constants
                                             We have picked up all the set constants within this set parentheses. Set Constant Value Descriptor is created.
                                           RETURN .SETVALPTR;
                                           END:
                               L1:10616
; Referenced LOCAL symbol LOW_RANGE_VAL is probably not initialized
```

			0	FFC	00000	GET_SET	CONSTANT		
50	8F	00000000	FF 30	91	00002 0000A		CMPB BNEQ	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 aCHARPTR, #93	1044
0000000G	7E 00	7A	30 20 8F 02	DD 9A FB	0000C 0000E 00012		PUSHL MOVZBL CALLS	2\$ #32 #122, -(SP) #2, DBG\$MAKE_SKELETON_DESC	1049
03 06 18 14	52 A2 A2	0608	50 01 8F	00 8E 80 9E	00019 00010 00020		MOVL MNEGB MOVW MOVAB	RO, SETVALPTR #1, 3(SETVALPTR) #1544, 6(SETVALPTR) 32(R2), 24(SETVALPTR)	1049 1049
12	A2	FFFF0020 00000000	8F EF 0181	DO D6 31	0002B 00033 00039	15:	MOVL INCL BRW	#-65504, 20(SETVALPTR) CHARPTR 22\$	1049 1050 1050
00000000	59 EF		01 58 01	D0 D4 D0	0003C 0003F 00041	1\$: 2\$:	MOVL	#1. CREATE THIS SUBSCR IS RANGE	1051 1051 1051
0000000	02	00000000	EF E8	01	00048 0004F	3\$:	CMPL BEQL	#1, TERMINATOR_CODE TERMINATOR_CODE, #2	1051
	58	00000000	ĒF	00	00051		MOVL	EXPRESSION_RADIX, SAVED_RADIX	: 1052

					1	-Sep-19	84 02:10 84 12:17	:13	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page 336 (42)
00000000.	EF	00000000.	OA EF 7E 02	DO 9F	00058 0005F 00065 00067		MOVL	W10.	EXPRESSION RADIX	; 1052 ; 1052
D42A	CF 53		65	FB	00067 0006C		CALLS	#2 . [RGSEXPRESSION PARSER	
00000000	ÉF	00000000	508FDF1F90F	094B0052	0006F 00076 0007C 0007E 00084 0008B		MOVL MOVL TSTL BNFQ	SAVED	ALPTR PRADIX, EXPRESSION_RADIX NATOR_CODE	1052
0000000G	00	00028E90	8F 01	DD FB	0007E		BNEQ PUSHL CALLS	#1675	668 IB\$SIGNAL	
00000000	EF 7E	00000000	EF 59	CO E9	00040	48:	CALLS ADDL2 BLBC	TERMI	68 IB\$SIGNAL NATOR_LENGTH, CHARPTR E, 9\$	1053 1054 1054
	7E	7A	20 8F	CEP DA B D P B B B B B B B B B B B B B B B B B	00099 0009F 000A6 000A9 000B1 000B7 000C0 000C4 000C8		MOATOL	#122	-(SP)	1054
0000000G	7E 00 52		50	DO	0009F		MOVL	RO. S	BETVALPTR	
03 06 14 18	A2	90000000G 8080	0200 0500 802 050 050 050 050 050 050 050 050 050 0	90 B0	000A9 000B1		MOVL MOVB MOVW MOVW MOVAB	#1544	BG\$MAKE_SKELETON_DESC SETVALPTR BB_LANGUAGE, 3(SETVALPTR) - 6(SETVALPTR) 20(SETVALPTR) 2), 24(SETVALPTR) ETVALPTR), SETVAL PTR), RO	1054 1054 1055 1055 1055 1055
18	A2 A2	20	A2	9E	000B7		MOVAB	32 (R2	20(SETVALPTR) 2), 24(SETVALPTR)	: 1055
	5A 50 02	20 18 06	AZ A3	DO 9A 91	000C0 000C4		MOVL	6(VAL	TVALPTR), SETVAL LPTR), RO	: 1055 : 1055
			50 1B	1F	000CB		CMPB BLSSU CMPB	RO. 4	12	: 1055
	03		16	91 1A	0000D		BGTRU	55	13	
	7E	0100	01 8F	DD 3C	000D2 000D4		PUSHL	#256	, -(SP)	1055
	7E 00	16	01 8F 08 A3	DD 3C DD 9A FB	000D0 000D2 000D4 000D9 000DB		PUSHL	#8 22 (VA	ALPTR), -(SP)	1055
0000000G			1A		000DF 000E6 000E8		CALLS BRB CMPB	6\$	BG\$TYPEID_FOR_SET	
	04		50 1B A3	91	000E8	5\$:	BNEQ	RO. 4		1056
	56 7E	0100	A3 8F	DO	000EB 000ED 000F1 000F6		MOVZWL	#256	PTR), TYPEID (SP)	: 1056 : 1056
		00	8F 08 A6 03	00	VVVE 8		PUSHL	#8 12(T)	(PEID)	: 1056
000000006	00 A2			FB	000FB 00102	6\$:	MOVL	#3, D	DBG\$TYPEID_FOR_SET	
		00028708	OD 8F	11 DD	000FB 00102 00106 00108 0010E 00115	75:	BRB PUSHL	#1658	348	: 1055 : 1056
0000000G	00		01 59	FB D4	0010E 00115	85:	CALLS	W1. L	IB\$SIGNAL E LPTR	1057 1058
			52	DD	00117	9\$:	PUSHL	VALPI	ALPTR	
000000006	00		02	DD FB	0011B 0011D		CLRL PUSHL PUSHL PUSHL CALLS BLBS PUSHL	#2	BG\$PERFORM_TYPEID_CHECK	1058
		00028F20	50 8F	E8	00124		BLBS PUSHL	RO. 1	U.S.	1058
0000000G	90 57	18	01 A3	FB	0012D 00134	10\$:	MOVI	#1. L	IB\$SIGNAL	: 1058
	OF	00000000	050 861 861 861 861 861 861	D1 12	00119 0011B 0011D 00124 00127 0012D 00134 00138		CMPL BNEQ	TERM!	NATOR_CODE, #15	1058
	OD	00028F08	58 8F	DB4DDDB8DB0129D	00141		CMPL BNEQ BLBC PUSHL	THIS	SUBSCR_IS_RANGE, 11\$	1059
0000000G	00		01	FB	00144 0014A		CALLS	#1, [.IB\$SIGNAL	:

DBGPARSER V04-000	L 7 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRCJDBGPARSER.B32;1	Page 337 (42)
	01 D0 00151 11\$: MOVL #1, THIS_SUBSCR_IS_RANGE 10 11 00154 BRB 14 11 58 E9 00156 12\$: BLBC SUBSCR_IS_RANGE, 14\$ 12 55 D1 00150 CMPL LOW_RANGE_VAL, R4 13	: 1059 : 1059 : 1061 : 1061
00000000G	58 D4 0016E 13\$: CLRL THIS_SUBSCR_IS_RANGE 06 11 00170 BRB 15\$	1061 1061 1062
00000100	54 67 DO 00172 14\$: MOVL (VALADDR), R4 55 54 DO 00175 MOVL R4, LOW_RANGE_VAL 55 D5 00178 15\$: TSTL LOW_RANGE_VAL 09 19 0017A BLSS 16\$ 8F 55 D1 0017C CMPL LOW_RANGE_VAL, #256 0D 15 00183 BLEQ 17\$	1063
00000000G	00028248 8F DD 00185 16\$: PUSHL #164424 00 01 FB 0018B CALLS #1, LIB\$SIGNAL 54 D5 00192 17\$: TSTL R4	1063
00000100 00000006	8F 54 D1 00196 CMPL R4, #256 0D 15 0019D BLEQ 19\$ 00028248 8F DD 0019F 18\$: PUSHL #164424 00 01 FB 001A5 CALLS #1, LIB\$SIGNAL 50 FF A5 9E 001AC 19\$: MOVAB -1(R5), I	1063
00 F8	50	1051 1064 1064

; Routine Size: 449 bytes, Routine Base: DBG\$CODE + 3162

ROUTINE GET_SUBSCRIPTS(PRIMPTR): NOVALUE =

FUNCTION This routine picks up subscript values in an array reference. It calls DBG\$EXPRESSION_PARSER to parse and evaluate each subscript expression. It also checks the data type of each subscript value and converts it to the appropriate data type as necessary. The ultimate subscript values are stored as integers in the Primary Descriptor Array Sub-Node for the array being subscripted. A new Sub-Node for the array element type is then appended so that the Primary Descriptor becomes a descriptor for the array element selected by the subscripting.

This routine also handles subscript ranges, such as ARR(1:5,3:10). It does so by modifying the subscript lower and upper bounds in the Array Sub-Node subscript vector to in effect define a new array, namely the array "slice" defined by the subscript ranges. In this case no Sub-Node for the array element is appended since the Primary Descriptor still defines an array.

This routine assumes that the opening subscript parenthesis has already been found and that the parse pointer points to the start of the first subscript expression. When this routine returns, the parse pointer is left pointing at the first character after the closing subscript parenthesis.

INPUTS PRIMPTR - A pointer to the Primary Descriptor for an array about to be subscripted.

The PRIMPTR Primary Descriptor is changed to include the subscript information (the actual subscript values) and a new Sub-Node for the selected array element. PRIMPTR itself is not changed, however.

BEGIN

PRIMPTR: REF DBG\$PRIMARY;

! Pointer to array Primary Descriptor

BITSIZE,
CHECK VAL,
DECLTYPE: REF DBG\$VALDESC,

DESCR: DBG\$STG_DESC, DSCADDR: REF DBG\$STG_DESC, FCODE,
LA PTR: REF VECTOR[,BYTE],
LOW RANGE VAL,
NODEPTR: REF DBGSPRIM_NODE,
SAVED RADIX,
SUBSCR_COUNT,
SUBVECTOR: REF DBG\$PRIM_NODE_SUBS, !

! Bit size of subscript value data type

Pointer to Value Descriptor for declared subscript data type String descriptor Pointer to a string descriptor
Data type FCODE for array element type
Lookahead pointer into input
Low value of a subscript range
Pointer to Prim Descr Array Sub-Node
Temporarily saved expression radix Actual subscript count in input line Pointer to subscript block-vector in Primary Descr Array Sub-Node

DESCR[DSC\$B_CLASS] = DSC\$K_CLASS_S; DESCR[DSC\$B_DTYPE] = DSC\$K_DTYPE_T;

:10672

Page 339 (43)

```
DB
```

Page 340

```
DBGPARSER
V04-000
                                                                                                  16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                      VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGPARSER.B32;1
                                                                         DESCREDSC$W_LENGTH] = .BITSIZE/8;
GET_SUBSTRING (.PRIMPTR, DESCR);
RETURN;
                         10764
10765
10766
10767
10673
10674
10675
10676
10677
10678
10680
10681
10683
10684
10685
                                                                         END:
                                                                   END:
                         10770
                                                                  .PRIMPTR[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_DESCR
                         10771
                                                             THEN
                         10772
                                                                   BEGIN
                         10773
                                                                   DBG$STA_TYP_DESCR (.PRIMPTR [DBG$L_DHDR_TYPEID], DSCADDR);
IF .DSCADDR[DSC$B_DTYPE] EQL DSC$K_DTYPE_T
                         10774
10775
10776
                                                                   THEN
                                                                         BEGIN
10686
10687
10688
                                                                         GET_SUBSTRING (.PRIMPTR, .DSCADDR); RETURN;
                         10777
                         10778
                         10779
                                                                         END:
10689
                         10780
                                                                   END:
                         10781
:10691
                         10782
10783
                                                                for typed pointers there are two special cases: for language (, subscripting of pointers is allowed; e.g., PTR[n] is equivalent to *(PTR+n) for ADA, the pointer dereference is implicit; thus if PTR is a pointer then PTR(I) in ADA is equivalent to PTR*[I] in PASCAL.
:10692
:10693
                         10784
:10694
                         10785
:10695
                         10786
:10696
                         10787
:10697
                         10788
:10698
                         10789
:10699
                         10790
                                                             IF .PRIMPTR[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_TPTR
                         10791
:10700
                                                             THEN
                         10792
10793
:10701
:10702
                                                                   CASE .DBG$GB_LANGUAGE FROM DBG$K_MIN_LANGUAGE TO DBG$K_MAX_LANGUAGE OF
                         10794
:10703
:10704
                         10796
10797
                                                                            For language C, subscripting of pointers is allowed; e.g.,
PTR[n] is equivalent to *(PTR+n)
:10705
:10706
                         10798
:10707
:10708
                         10799
                                                                         DBG$K C]:
:10709
                         10800
:10710
                         10801
                                                                               LOCAL
:10711
                                                                                     ADDRESS,
                                                                                                                             Address of array (value of pointer)
:10712
                                                                                     DUMMY, ! Dummy output parameter VALPTR: REF DBG$VALDESC;! Pointer to a Value Descriptor
:10713
                         10804
:10714
                         10805
:10715
                         10806
:10716
                                                                               BUILTIN
:10717
                         10808
                                                                                     REMQUE:
:10718
                         10809
:10719
10720
10721
10722
10723
10724
10725
                                                                                  Compute the value of the pointer that is represented
                         10812
10813
                                                                                  by the current primary.
                                                                               DBGSPRIM_TO_VAL (.PRIMPTR, DBGSK_VALUE_DESC, VALPTR);
                         10814
                         10815
                                                                               ADDRESS = . VALPTR[DBG$L_VALUE_VALUE0];
                         10816
                         10817
;10727
;10728
                         10818
                                                                                  Unlink the existing subnode and build a new one to
                         10819
                                                                                  describe an array.
:10729
```

```
DBGPARSER
V04-000
                                                                                       16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                       VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.832:1
                                                                      NODEPTR = .PRIMPTR[DBG$L_PRIM_BLINK];
REMQUE(.NODEPTR, DUMMY);
TYPEID = DBG$TYPEID_FOR_ARRAY(.PRIMPTR[DBG$L_DHDR_TYPEID],
                                                                      DBG$BUILD_PRIMARY_SUBNODE (.PRIMPTR, RST$K_DATA, 0,

RST$K_TYPE_ARRAY, .TYPEID, 0);

NODEPTR = .PRIMPTR[DBG$L_PRIM_BLINK];
                                                                      NODEPTREDBG$L_PNODE_RELOC] = .ADDRESS;
                                                                   for ADA, the pointer dereference is implicit; thus if PTR is a pointer then PTR(I) in ADA is equivalent to PTR^[I] in PASCAL.
                                                                 [DBG$K ADA]:
                                                                        Dereference the pointer.
                                                                      WHILE .PRIMPTR[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_TPTR DO GET_DEREFERENCE(.PRIMPTR);
                                                                      ! If the Primary is still not an array then
                                                                         signal an error.
                                                                      IF .PRIMPTR[DBG$B_DHDR_FCODE] NEQ RST$K_TYPE_ARRAY
                                                                           SIGNAL (DBG$_NOTARRAY);
                                                                      END:
                                                                 ! for other languages, we cannot subscript typed pointers.
                                                                [INRANGE]:
                                                                      SIGNAL (DBG$_NOTARRAY);
                                                                   We do not expect any other language codes.
                                                                 [OUTRANGE]:
                                                                      $DBG_ERROR('DBGPARSER\GET_SUBSCRIPTS 5');
                                                                TES;
                                                           END
                                                        Else the variable is neither a string nor a typed pointer
:10780
                                                        nor an array so subscripting is not allowed.
:13781
                                                           SIGNAL (DBG$_NOTARRAY);
:10785
:10786
```

Page 341 (43)

:10843

TOKEN = DBG\$LEXICAL SCANNER (FALSE, FALSE, SUBSCRIPT TERM TBL, 0); IF . TOKEN NEG TERMINATOR_TOREN THEN BEGIN LOCAL ASCIC_STRING: VECTOR[2,BYTE];

```
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                                                                            VAX-11 Bliss-32 V4.0-742
[DEBUG.SRC]DBGPARSER.B32:1
                                                                                                      ASCIC_STRING[0] = 1;
ASCIC_STRING[1] = .CHARPTR[0];
SIGNAT(DBG$_SYNERREXPR, 1, ASCIC_STRING);
10844
10845
10846
10847
10848
10849
10851
10851
10853
10854
10856
10857
10858
10859
10860
                                                                                              IF .TERMINATOR_CODE EQL TOKENSK_TERM_COLON THEN
                                                                                                     SIGNAL (DBG$ INVRANSPEC);
.TERMINATOR_CODE EQL TOKENSK_TERM_NONE
                                                                                              SIGNAL (DBG$ MISCLOSUB);
CHARPTR = .CHARPTR + .TERMINATOR_LENGTH;
                                                                                                  Turn this reference into a range.
                                                                                                  If it was not already a range, then turn all previous
                                                                                                  subscripts into ranges.
  10861
10862
10863
                                                                                               IF NOT .NODEPTREDBG$V_PNARR_RANGE]
                                                                                                       BEGIN
  10863
10864
10865
10866
10867
10868
10870
10871
10872
10873
                                                                                                      NODEPTR[DBG$V_PNARR_RANGE] = TRUE;
INCR I FROM O TO .SUBSCR_COUNT - 1 DO
                                                                                                               SUBVECTOR[.I, DBG$L_PNSUB_LBOUND] =

.SUBVECTOR[.I, DBG$L_PNSUB_UBOUND] =

SUBVECTOR[.I, DBG$L_PNSUB_UBOUND] =

.SUBVECTOR[.I, DBG$L_PNSUB_SVALUE];
                                                                                                               END:
                                                                                             END;
SUBSCR_COUNT = .SUBSCR_COUNT + 1;
NODEPTR[DBG$v_PNARR_RANGE] = TRUE;
  10874
10875
10876
10877
                                                                                     ELSE
                                                                                              BEGIN
10880
10881
10882
10883
10884
10885
10886
10889
10890
10891
10893
10893
10894
10897
10898
10898
10899
                                                                                                  Call the expression parser to pick up the next subscript expression
                                                                                                  and its value. Note that we set the radix to decimal over this call and then restore it. Also note that the Expression Parser sets TERMINATOR_CODE and TERMINATOR_LENGTH as a side-effect.
                                                                                              SAVED_RADIX = .EXPRESSION_RADIX;

EXPRESSION_RADIX = DBG$K_DECIMAL;

VALPTR = DBG$EXPRESSION_PARSER (FALSE, .SUBSCRIPT_TERM_TBL);

EXPRESSION_RADIX = .SAVED_RADIX;
                                                                                                  Check the terminator code. If there was no terminator (the input line just ended), signal an error. Otherwise we got a comma or closing subscript parenthesis and we increment CHARPIR to get past it.
                                                                                              if .TERMINATOR_CODE EQL TOKEN$K_TERM_NONE THEN SIGNAL(DBG$_MISCLOSUB);
CHARPTR = .CHARPTR + .TERMINATOR_LENGTH;
```

```
We now need to convert the subscript to one of the appropriate dtype. We need to set up a target descriptor for the conversion routine. We allocate a skeleton descriptor and fill in some of
                                                                                                              the fields.
                                                                                                        DECLTYPE = DBG$MAKE_SKELETON_DESC(DBG$K_VALUE_DESC, 4);
DECLTYPE[DBG$B_DHDR_KIND] = RST$K_DATA;
TYPEID = .SUBVECTOR[.SUBSCR_COUNT, DBG$L_PNSUB_TYPEID];
DECLTYPE[DBG$L_DHDR_TYPEID] = .TYPEID;
DECLTYPE[DBG$L_VALUE_POINTER] = DECLTYPE[DBG$A_VALUE_ADDRESS];
                                                                                                              We now must fill in the FCODE, CLASS, DTYPE, and LENGTH fields
 10914
                                                                                                              of the target descriptor. To do this, we need to look at the
   10915
                                                                                                              typeid.
  10917
                                                                                                          IF .TYPEID EQL O
  10918
                                                                                                          THEN
  10919
                                                                                                                   BEGIN
                                                                                                                       No typeid available - assume longword integer.
                                                                                                                  DECLTYPE[DBG$B_DHDR_FCODE] = RST$K_TYPE_ATOMIC;
DECLTYPE[DBG$B_VALUE_CLASS] = DSC$K_CLASS_S;
DECLTYPE[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
DECLTYPE[DBG$W_VALUE_LENGTH] = 4;
                                                                                                         ELSE
                                                                                                                   BEGIN
                                                                                                                    ! Typeid is available - determine FCODE from it.
  10936
10937
10938
10939
                                                                                                                  DBG$STA_SYMSIZE(.TYPEID, BITSIZE);
FCODE = .TYPEID[RST$B_FCODE];
DECLTYPE[DBG$B_DHDR_FCODE] = .FCODE;
IF .FCODE EQL RST$K_TYPE_ATOMIC
  10940
10941
10942
10943
10944
                                                                                                                   THEN
                                                                                                                            BEGIN
                                                                                                                                 Atomic data - determine class, dtype, and length from DST.
 10945
10946
10947
10948
10949
10950
10951
10952
                                                                                                                             DBG$STA_TYP_ATOMIC(.TYPEID, TYPECODE, BITSIZE);
IF .TYPECODE EQL DST$K_BOOL
                                                                                                                             THEN
                                                                                                                            TYPECODE = DSC$K_DTYPE_TF;

DECLTYPE[DBG$B_VALUE_CLASS] = DBG$MAP_DTYPE_CLASS (
    TYPECODE, FALSE);

DECLTYPE[DBG$B_VALUE_DTYPE] = .TYPECODE;

IF .TYPECODE REQ DSC$K_DTYPE_V

AND .TYPECODE NEQ DSC$K_DTYPE_SV

AND .TYPECODE NEQ DSC$K_DTYPE_VU

AND .TYPECODE NEQ DSC$K_DTYPE_SVU

AND .TYPECODE NEQ DSC$K_DTYPE_SVU

AND .TYPECODE NEQ DSC$K_DTYPE_TF
10954
10955
10956
10957
                                      11046
11047
11048
```

```
DBGPARSER
V04-000
                                                                                                                                                                     16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                                                                                   VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
10958
10960
10961
10963
10963
10963
10965
10965
10966
10967
10973
10973
10973
10973
10976
10976
10981
10983
10983
10983
10983
10983
10983
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
10993
                                                                                                                                      THEN
                                                                                                                                                 DECLTYPE[DBG$W_VALUE_LENGTH] = (.BITSIZE+7)/8
                                                                                                                                      ELSE
                                                                                                                                                 DECLTYPE[DBG$W_VALUE_LENGTH] = .BITSIZE;
                                                                                                                            ELSE IF .FCODE EQL RSTSK_TYPE_DESCR
                                                                                                                                      BEGIN
                                          11059
                                                                                                                                           Descriptor data - determine class, dtype, and length from DST.
                                          11060
                                          11061
                                                                                                                                      DBG$STA_TYP_DESCR (.TYPEID, DECLTYPE[DBG$A_VALUE_VMSDESC]);
                                                                                                                                       IF .DECETYPEEDBG$B_VALUE_DTYPE] EQL DST$K_BOOL
                                          11063
                                          11064
                                                                                                                                                DECLTYPE[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_TF;
                                          11065
                                         11066
                                         11067
                                                                                                                            ELSE
                                                                                                                                      BEGIN
                                          11069
                                          11070
                                                                                                                                           Language-specific fcodes. Here we dummy in the dtype field
                                          11071
                                                                                                                                           with a special code. Determine class and length information
                                          11072
                                                                                                                                           using routines in DBGEVALOP.
                                          11074
11075
                                                                                                                                      DECLTYPE[DBG$B_VALUE_CLASS] = 0;
DECLTYPE[DBG$B_VALUE_DTYPE] = 0;
DECLTYPE[DBG$W_VALUE_LENGTH] = (.BITSIZE+7)/8;
                                         11076
                                          11077
                                                                                                                                      END:
                                          11078
11079
                                                                                                                            END:
                                          11080
                                          11081
                                                                                                                      finally call the conversion routine. This routine checks that
                                          11082
11083
                                                                                                                      the conversion is legal before doing it.
                                          11084
                                                                                                                 VALPTR = DBG$EVAL_LANG_OPERATOR(DBG$GL_CONVERT_TOKEN, .VALPTR, .DECLTYPE);
                                          11085
                                          11086
                                          11087
                                                                                                                      Check that the subscript value is within the array bounds and give an informational message if not (execution continues after the message).
                                          11088
                                          11089
                                                                                                                      for ADA, we make sure to check enumeration subscripts by their
                                          11090
                                                                                                                      position and not by their value.
                                          11091
                                          11092
11093
                                                                                                                  VALADDR = .VALPTR[DBG$L_VALUE_POINTER]:
                                                                                                                         (.DBG$GB_LANGUAGE EQE DBG$R_ADA) AND
                                                                                                                          (.FCODE EQL RSTSK_TYPE_ENUM)
                                          11096
11097
                                                                                                                           CHECK_VAL = DBGSENUM_POS(.TYPEID,.VALADDR[0])
                                          11098
11099
                                                                                                                 ELSE
                                                                                                                            CHECK_VAL = .VALADDR[0];
                                          11100
                                          11101
   11010
                                                                                                                 IF (.CHECK_VAL LSS .SUBVECTOR[.SUBSCR_COUNT, DBG$L_PNSUB_LBOUND]) OR (.CHECK_VAL GTR .SUBVECTOR[.SUBSCR_COUNT, DBG$L_PNSUB_UBOUND])
:11011
                                          11102
 :11012
                                                                                                                 THEN
 :11013
                                                                                                                            SIGNAL (DBG$_SUBOUTBND, 4, .SUBSCR_COUNT + 1, .CHECK_VAL
 :11014
                                                                                                                                                                     .SUBVECTOR[.SUBSCR_COUNT, DBG$L_PNSOB_LBOUND],
```

Page 345 (43) D

D

11160 11161 11162

```
8
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                              VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32:1
```

.SUBVECTORE.SUBSCR_COUNT, DBG\$L_PNSUB_UBOUND]);

If the terminator at the end of this subscript expression was a colon we have a subscript range (for example, 'ARR(1:5,2)''). We thus set the subscript-range flag and save the low value of the range, i.e. the value we just picked up. If this is the first range in the subscript list, we also turn all previous subscripts into ranges by setting the lower and upper bound for each such subscript to the corresponding subscript value. This in effect defines a new array which constitutes a "slice" of the original array. .TERMINATOR_CODE EQL TOKENSK_TERM_COLON

THEN BEGIN IF .THIS_SUBSCR_IS_RANGE THEN SIGNAL(DBG\$_INVRANSPE();
THIS_SUBSCR_IS_RANGE = TRUE;
LOW_RANGE_VAL = .VALADDR[0];
IF_NOT .NODEPTR[DBG\$V_PNARR_RANGE] THEN NODEPTREDBGSV_PNARR_RANGE] = TRUE; INCR I FROM O TO .SUBSCR_COUNT - 1 DO

SUBVECTOR[.I, DBG\$L PNSUB LBOUND] =
.SUBVECTOR[.I, DBG\$L PNSUB_SVALUE];
SUBVECTOR[.I, DBG\$L PNSUB UBOUND] =
.SUBVECTOR[.I, DBG\$L_PNSUB_SVALUE]; END:

END:

END

The terminator was not a colon, so we now have the full subscript specification. Fill the subscript value into the Array Sub-Node's subscript vector. Set up the bounds for an array "slice" if this or any previous subscript specification in this array reference consisted of a subscript range. Also bump the subscript count.

ELSE BEGIN

> If this subscript is specified as a subscript range, check that the first value in the range is not greater than the second. Also clear the subscript-is-range flag for the next subscript. IF .THIS_SUBSCR_IS_RANGE THEN BEGIN IF .LOW_RANGE_VAL GTR .VALADDR[0] THEN SIGNAL(DBG\$_INVRANSPEC);
> THIS_SUBSCR_IS_RANGE = FALSE;

NODEPTR[DBG\$V_PNODE_EVAL] = TRUE;

```
VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                     Otherwise, set the low range value to be the subscript value.
                         LOW_RANGE_VAL = .VALADDR[0];
                     If this or any previous subscript in this array reference contained a range specification (as in ARR(5:10)), then we arrange the array's lower and upper bounds to reflect the array "slice" the user is requesting.
                   IF .NODEPTR[DBG$V_PNARR_RANGE]
                         SUBVECTOR[.SUBSCR_COUNT, DBG$L_PNSUB_LBOUND] = .LOW_RANGE_VAL;
SUBVECTOR[.SUBSCR_COUNT, DBG$L_PNSUB_UBOUND] = .VALADDR[0];
                     finally fill in the subscript value itself (the start of the
                     range), increment the subscript count, and loop.
                   SUBVECTOR[.SUBSCR_COUNT, DBG$L_PNSUB_SVALUE] = .LOW_RANGE_VAL;
SUBSCR_COUNT = .SUBSCR_COUNT + 1;
                                                         ! End of WHILE loop over subscripts
  We have picked up all the subscripts within this set of subscript parentheses. Now check that the subscript count is the same as the dimension count unless the language allows fewer subscripts (as in Pascal where the array reference X[2,3][4] is valid). If fewer subscripts are allowed
  we return now, leaving the Array Sub-Node at the end of the Primary.
NODEPTR[DBG$B_PNARR_SUBCNT] = .SUBSCR_COUNT;
IF .SUBSCR_COUNT NEW . NODEPTREDBG$B_PNARR_DIMENT]
     IF .MULTIPLE_SUBSCR THEN RETURN;
SIGNAL(DBG$_TOOFEWSUB, 1, .NODEPTR[DBG$B_PNARR_DIMCNT]);
  If any subscript range was specified, we leave the Array Sub-Node at the end of the Primary even if all subscripts were specified. In other words, we still have an array, namely the specified "array slice".
IF .NODEPTR[DBG$V_PNARR_RANGE] THEN RETURN;
  All the subscripts are specified. Now set the EVAL bit in the Array
   Sub-Node to indicate that subscripting actually is being done. Also
   construct a new Sub-Node for the array element type and append it to
```

DBGPARS V04-000 :11129 :11130 :11131 :11132 :11133 :11134	ER		112 112 112 112 112	20 22 23 22 25	222221			FCC DBG RET	UNIA,	= DBG ILD_P	\$ST RIM	A TYP	EFCO UBNO			984 02:10 984 12:17 8G\$L PNAR 8ST\$R DAT NODEPTRE	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1 R_CELLTYPE]); A, 0, DBG\$L_PNARR_CELLTYPE], 0);	Page 348 (43)
5F 54 5F 54	45 45 30	47 35 47 31	5C 20 5C 20	52 53 53 53	45454	530	52 49 52 49	41 52 41 52	50 43 50 43	47 53 47 53	4222	44 55 44 55	1A 53 1B 53	03398 033A7 033B3 033C2	P.AYL: P.AYM:	.PSECT .ASCII	DBG\$PLIT,NOWRT, SHR, PIC,0 <26>\DBGPARSER\<92>\GET_SUBSCRIPTS 5\ <27>\DBGPARSER\<92>\GET_SUBSCRIPTS 10\	
				20	AE 50	000	00000	9 06 06 06	55208F 005301 02 008 7E 00 AE AE AE 03	00028	CO 04 02 7E8 042 02 2C4 08 30 1C7A 200 10E 34 02	01 01 01 01 01 01 01 01 01 01 01 01 01 0	OFFC 9E00913DBE912199FDB12FCD91312070891191	00000 00000 000006 000016 000018 000015 000029 000029 000038 000038 000041 000048 000048 000057 000057 000068 000068 000074 000077 000077 000088	1\$: 2\$:	PSECT SCRIPTS: WORD MOVAB MOVL CMPB BEGLL CALAB CMPB BNEQ BNEQ BNEQ BNEQ BNEQ BNEQ BNEQ BNE	DBG\$CODE,NOWRT, SHR, PIC.O Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 -64(SP), SP PRIMPTR, R2 R2, DBG\$GL_CURRENT_PRIMARY 2(R2), #12T 1\$ #165864 #1, LIB\$SIGNAL 4(R2), R3 2(R3), #1 2\$ BITSIZE TYPECODE 8(R2) #3, DBG\$STA_TYP_ATOMIC TYPECODE, #24 3\$ VALPTR #122, -(SP) R2 #3, DBG\$PRIM_TO_VAL #20, VALPTR, DSCADDR DSCADDR, RO 2(R0), #14 4\$ TYPECODE, #14 4\$ #270, DESCR+2 #8, BITSIZE, RO RO, DESCR DESCR 6\$ 2(R3), #3	1065 1071 1072 1072 1072 1073 1074 1074 1075 1075 1075 1076 1076 1076 1076

DBGPARSER V04-000						1	Sep-	984 02:10 1984 12:17	0:13 VAX-11 Bliss-32 V4.0-742 7:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 34
	0000000G	00 50 0E	20 08 20 02	21 A22 A02 A0A 5522	12 9F DD FB DO	00091 00094 00098		BNEQ PUSHAB PUSHL CALLS MOVL CMPB BNEQ PUSHL PUSHL CALLS RET CMPB BEQL BRW CASEB WORD	7\$ DSCADDR 6(R2) #2, DBG\$STA_TYP_DESCR DSCADDR, R0 2(R0), #14	107
	00004		02	50 50 50	12 00 00	0009F 000A3 000A5 000A7	5\$: 6\$:	BNEQ PUSHL PUSHL	RO R2	107
	0000v	CF 06	02	A3	FB 04 91 13	000AF 000AF 000B3	7\$:	RET CMPB BEQL	2(R3). #6	107 107
008E 008i 002D 008i 008i	E O	00 08E 08E 079	0000000G	0096	31 8F	000A9 000AE 000AF 000B3 000B5 000C0 000C8 000D0	8\$: 9\$:	CASEB .WORD	8\$ 13\$ DBG\$GB_LANGUAGE, #0, #10 13\$-9\$,- 13\$-9\$,- 13\$-9\$,- 13\$-9\$,- 13\$-9\$,- 13\$-9\$,- 13\$-9\$,-	107
	0000000G	00	00000000° 00028362	EF 01 8F 03	9F DD DD FB	000D6 000DC 000DE 000E4 000EB		PUSHAB PUSHL PUSHL CALLS BRB PUSHAB	11\$-9\$,- 13\$-9\$ P.AYL #1 #164706 #3, LIB\$SIGNAL 14\$	108
	00000000G	7E 00 50 53 56 50	24 7A 24 20 18	6EEF23E0A0265A220	Ur	000ED 000F0 000F4 000F6 000FD 00101 00105 00109	10\$:	PUSHAB MOVZBL PUSHL CALLS MOVL MOVL MOVL REMQUE	VALPTR	108 108 108 108 108 108
	0000000G	52 00 58	04 08	AC A2 02 50	DD DD FB DO	0010C 0010E 00112 00115 0011C		PUSHL MOVL PUSHL CALLS MOVL	ADDRESS PRIMPTR, R2 8(R2) #2, DBG\$TYPEID_FOR_ARRAY R0, TYPEID -(SP)	
	CC1A 14	7E CF 56 A6 06	18 02	581 052 052 052 052 052 052 052 052 052 052	11 91	00116 00117 00121 00123 00125 00128 00126 00137 00137 00137 00136	115:	MOVZBL PUSHL CALLS MOVL MOVL REMQUE PUSHL PUSHL PUSHL PUSHL PUSHL PUSHL MOVQ PUSHL CALLS MOVL CALLS MOVL CALLS CMPB BNEQ PUSHL CALLS	#122, -(SP) R2 #3, DBG\$PRIM_TO_VAL VALPTR, R0 32(R0), ADDRESS 24(R2), NODEPTR (NODEPTR), DUMMY ADDRESS PRIMPTR, R2 8(R2) #2, DBG\$TYPEID_FOR_ARRAY R0, TYPEID -(SP) TYPEID #1 #6, -(SP) R2 #6, DBG\$BUILD_PRIMARY_SUBNODE 24(R2), NODEPTR ADDRESS, 20(NODEPTR) 14\$ 2(R3), #6 12\$ R2 #1, GET_DEREFERENCE	108 108 108 108 108 107 108
	F91E	CF		A3 09 52 01	DD FB	0013D 0013F 00141		BNEQ PUSHL CALLS	R2 #1, GET_DEREFERENCE	108

23\$:

DD

BNEQ

PUSHL

#167568

00000000

00028E90

1094

1094

					1	6-Sep-	1984 02:10 1984 12:17	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page 351 (43)
25	000000000 000000000 02	00 EF 6A AA 51	00000000° 01 13 08 01	FB CO E8 CE	00249 00254 00258 00250	24\$:	CALLS ADDL2 BBS BISB2 MNEGL	#1, LIB\$SIGNAL TERMINATOR_LENGTH, CHARPIR #19, (R10), 27\$ #8, 2(R10) #1, I	1094 1095 1095 1095
50		51	08 A042 6042	C5 9F 9F	0025F 00261 00265 00269	25\$:	BRB MULL3 PUSHAB PUSHAB	26\$ #20, I, R0 8(R0)[SUBVECTOR] (R0)[SUBVECTOR]	1095
		9E	0C A042 6042	00 9F 9F	0026C 0026F 00273		MOVL	a(SP)+, a(SP)+ 12(R0)fSURVECTORI	1096
E4		9E 51	9E 59 59	PO F2	00276	26\$: 27\$:	PUSHAB MOVL AOBLSS INCL BISB2	(RÓ)[SUBVECTOR] a(SP)+, a(SP)+ SUBSCR_COUNT, I, 25\$ SUBSCR_COUNT #8, 2(R10) 16\$	1095 1096
	02	AA	08 FF0E	D6 88 31	0027b 0027f 00283		BISB2	#8, 2(R10)	1096
	000000000	AE EF	00000000' EF	00000	00286 0028E 00295	28\$:	BRW MOVL MOVL PUSHL	EXPRESSION_RADIX, SAVED_RADIX #10, EXPRESSION_RADIX SUBSCRIPT_TERM_TBL -(SP)	1097 1097 1098
	D033	CF	02	FB	0029D		CLRL	#2, DBGSEXPRESSION_PARSER	
	000000000.	CF AE EF	00000000 AE	DO D5 12	002A6 002AE 002B4		MOVL MOVL TSTL BNEQ PUSHL	RO, VALPTR SAVED RADIX, EXPRESSION_RADIX TERMINATOR_CODE 29\$	1098 1098
	000000000 000000000	00 EF	00000000' FF 020 00000000' EF 00000000' EF 00000000' EF 00000000' EF 00000000' EF 00000000' EF 01000000' EF 0100000' EF 0100000' EF 01000000' EF 0100000' EF 0100000' EF 0100000' EF 010000' EF 0100000' EF 010000' EF 01000' EF	FB CO	002B6 002BC 002C3	29\$:	ADDL2	#167568 #1, LIB\$SIGNAL TERMINATOR_LENGTH, CHARPTR	1098
		7E	7A 8F	DD 9A			PUSHL MOVZBL	#122, -(SP)	1099
	0000000G	7E 00 54	50	FB DO	002DB		MOVL	#2, DBG\$MAKE_SKELETON_DESC RO, DECLTYPE	
55	07	A4 59	06 14	90 C5	002DE		MOVL MOVB MULL3 PUSHAB	#6. 7(DECLTYPE)	1099
			10 A542	9F DO DO	002E6		PUSHAB	#20, SUBSCR COUNT, R5 16(R5)[SUBVECTOR] a(SP)+, TYPEID	
	08 18	58 A4	20 86	DO	002ED		MOVAR	TYPEID. 8(DECLTYPE)	1100
	10	A4 53	14 44	9E	002F6		MOVAB	TYPEID, 8(DECLTYPE) 32(R4), 24(DECLTYPE) 20(DECLTYPE), R3 TYPEID	1101
			00	9E 9E 05 12	002FC		MOVL MOVAB MOVAB TSTL BNEQ MOVB MOVL BRB PUSHAB	303	:
	06	63	01080004 8F	90 11	00302		WOAR	#17301508, (R3)	: 1101
			2C AE		00309 0030B	30\$:	BRB PUSHAB	BITSIZE	1100
	000000006	00	58 02	DD FB	0030E 00310		PUSHL CALLS MOVZBL	TYPEID #2, DBG\$STA_SYMSIZE	
	06	00 5B A4 02	18 A8	9F DD FB 9A	00317 0031B		MOVZBL	24(TYPEID), FCODE FCODE, 6(DECLTYPE)	1102 1102 1103
		02	5B 53	D1	0031F		CMPL BNEQ	#2. DBG\$STA_SYMSIZE 24(TYPEID), FCODE FCODE, 6(DECLTYPE) FCODE, #2 34\$	1103
			2C AE	12 9F 9F	00324		PUSHAB	TABECODE	1103
	00000000G	00	58	DD FB	00324 00327 0032A 0032C 00333		MOVB CMPL BNEQ PUSHAB PUSHAB PUSHL CALLS CMPL	TYPEID #3. DBG\$STA_TYP_ATOMIC TYPECODE, #T58	
	0000009E	8F	30 ĂĒ	DI	00333		CMPL	TYPECODE, #T58	: 1103

						N 16-S 14-S	8 ep-1984 02:10 ep-1984 12:17	:13	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page 352 (43)
	30	AE 57	34	04 28 7E AE	D4 00	33B 33D 341 319	BNEQ MOVL CLRL MOVL	31\$ #40, -(SP TYPE	TYPECODE CODE, R7	1104 1104 1104
	00000000G 03 02	00 A3 A3 01		027A5055515050505052A251550A120A0055	90 00 01 00	354 358	BNEQ MOVL CLRL MOVI PUSHL CALLS MOVB CMPL BEQL CMPL BNEQ BNEQ	R7207727272727373738383838383838383838383838	DBG\$MAP_DTYPE_CLASS 3(R3) 2(R3) #1	1104 1104
		29		57 0F	D1 00	35B 35D 360	EEQL CMPL BEQL	32\$ R7.	#41	1104
		22		57 0A	D1 00	362 365	CMPL BEQL	R7.	#34	1104
		2A		57 05	D1 00 13 00	367 36A	CMPL BEQL	R7,	#42	1104
		28	•	26	D1 00 12 00	36C 36F 371 329	BNEQ	36\$	#40	1104
		63 03	20	2C 58 18	11 00	375 339	S: MOVW S: BRB S: CMPL BNEQ	FCOD	IZE, (R3) E, #3	1105 1103 1105
	00000000G 9E	00 8F	02	53 58 02	DD 00 DD 00 FB 00 91 00	377 349 374 370 376 380 387	S: BRB S: CMPL BNEQ PUSHL PUSHL CALLS CMPB BNEQ MOVB	35\$ R3 TYPE #2,	ID DBG\$STA_TYP_DESCR	1106
	02	A3		15	12 00 90 00	38C 38E	BNEQ MOVB	37\$	DBG\$STA_TYP_DESCR), #158 2(R3)	1106
50 51	20	AE 50 63	02	0F A3 07 08	11 00 B4 00 C1 00	392 394 359 397 369	S: CLRW S: ADDL3 DIVL3	2(R3 #7,	BITSIZE, RO	1105 1107 1107
			000000000	51 54 AE EF	BO 00 DD 00 PF 00	39C 3AO 3A3 379 3A5 3A8	S: PUSHL PUSHL	R1,	(R3)	1108
	00000000G 1C 0C	OO AE AE O9	00000000G	AEF30000E8AB5E58208E532	FB 00 D0 00 D0 00 91 00	3A5 3A8 3AE 3B5 3B9 3C7 3C7	PUSHAB CALLS MOVL MOVL CMPB BNEQ TSTL BEQL CMPL BNEQ MOVL PUSHL PUSHL PUSHL CALLS MOVL BRB MOVL BRB	#3. R0. 24(R DBG\$	TR GL_CONVERT_TOKEN DBG\$EVAL_LANG_OPERATOR VALPTR O), VALADDR GB_LANGUAGE, #9 ID	1109 1109
				58	12 00 05 00	3C7	TSTL	TYPE	10	1109
		04		5B	D1 00 12 00	3CB	CMPL	204	E. #4	1109
		53	00	BÉ	00 00	3DO 3D4	MOVL	aval R3	ADDR, R3	1109
	90000000G	00 AE		58 02 50	DO 00 DD 00 FB 00 DO 00 11 00	3CB 3CE 3D0 3D4 3D8 3D5 3E5 3E5 3E9 3F1 3F7	PUSHL CALLS MOVL	TYPE	ID DBG\$ENUM_POS CHECK_VAC	
	08	53 AE	00	BE	DO 00	3E5 389	S: MOVL	PANT	ADDR, R3	1109
	08	9E	08 A	542 AE 0A 542	DO 00 DO 00 9F 00 D1 00	3ED 399	S: PUSHAB CMPL BLSS	8 (RS CHEC	ADDR, R3 CHECK_VAL)[SUBVECTOR] K_VAL, @(SP)+	1110
			OC A	542	19 00 9F 00	3F7	PUSHAB	12(R	5)[SUBVECTOR]	: 1110

59

18

						1	Sep-	1984 02:10 1984 12:17	:13	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Pag	e 353 (43)
		9E	00	AE	D1 15	003FB 003FF		CMPL	CHECK.	_VAL, a(SP)+	•	
			OC.	A542	15 9F	003FF 00401	40\$:	BLEQ PUSHAB PUSHAB PUSHAB PUSHL PUSHAB	415	CSUBVECTOR		1110
				9E	DD 9F	00405		PUSHL	a(SP)	SUBVECTOR]	:	
				A542	DD	00407 0040B		PUSHAB	a(SP)	LSUBVECTORJ		1110
			10		DD 9F	0040D		PUSHL	CHECK	VAL SCR_COUNT)		1110
				AE904F66F3EF	DD	00410 00413 00415		PUSHL	#4			
	0000000G	00	0002868B	8F	DD FB	00415 0041B		PUSHL	#1655	15 IB\$SIGNAL		
	00000000	03	00000000	' EE	D1	00422	415:	CMPL	TERMI	NATOR_CODE, #3		1111
		00	10	45 AF	12 F9	00429 0042B		CMPL BNEQ BLBC PUSHL	THIS	SURCED IS DANGE 426		1112
	00000000		00028F08	8F	E9	0042F 00435		PUSHL	#1676	BB IB\$SIGNAL HIS_SUBSCR_IS_RANGE DW_RANGE_VAL (RTO), 50\$ (R10)	:	
	00000000G	00 AE		01	FB	00435 0043C	425:	MOVL	#1: Ti	IB\$SIGNAL HIS SUBSCR IS RANGE	•	1112
41		AE 6E		53 13 08 01	DO	00440		MOVL	R3, L	DW RANGE VAL		1112 1112 1112 1112 1112
61	02	6A AA		08	E0	00443		BBS BISB2 MNEGL	#8. 2	(R10), 50\$ (R10)	•	1112
		51		01	ÇĘ	0044B		MNEGL	WIA A			1112
50		51		18	C5	0044E 00450	438:	BRB MULL3	44\$ #20.	I. RO		1113
			08	A042	9F 9F	00454		MULL3 PUSHAB PUSHAB	8(R0)	[SUBVECTOR]		1113
		9E		9E	DO	0045R		MOVL	a(SP)	I, RO ESUBVECTOR] SUBVECTOR] +, a(SP)+ DÉSUBVECTOR] SUBVECTOR] +, a(SP)+ R_COUNT, I, 43\$		
			00	A042	9F	0045E		PUSHAB	12(R0)	CSUBVECTOR)		1113
		9E 51		9E	ĎÔ	0045E 00462 00465		MOVL	a(SP)	+, a(SP)+		
E4		51		59 3A	F2	00468 00460	445:	AOBLSS BRB	SUBSCI	R_COUNT, 1, 43\$		1112
		17 53	10	AE	E9	0046E	458:	BLBC	THIS !	SUBSCR_IS_RANGE, 47\$		1115
		55		AE 6E 0D	D1 15	00472		CMPL	LOW_R	SUBSCR_IS_RANGE, 47\$ ANGE_VAL, R3		1115
	00000000	-	00028F08	8F	DD	00477		BLEQ PUSHL	#16768		;	
	0000000G	00	10	AE	FB D4	00477 00470 00484	465:	CALLS	THÍS	IB\$SIGNAL SUBSCR_IS_RANGE DW_RANGE_VAL	The same	1115
				AE 03 53	11	00487 00489	170	BRB	48\$	NI BANCE MAI	:	1115
OF		6E 6A		13	DO E1	00480	485:	MOVL BBC	#19. LI	(RTO), 49\$		1116
		9E	08 04	A542	9F	00490		PUSHAB	8(R5)	SUBVECTOR]		1117
			ÖČ	A542	DO 9F	00480 00490 00494 00498		PUSHAB	12(R5)	(RTO), 49\$ [SUBVECTOR] ANGE VAL, a(SP)+ (SP)+ SUBVECTOR] ANGE VAL, a(SP)+ SUBVECTOR] ANGE VAL, a(SP)+		1117
		9E		6542	DO 9F	0049C 0049F	404.	MOVL PUSHAB	R3, a	(SP)+		1118
		9E	04	AE 59	ĎÔ	004A2 004A6	470.	MOVL	LOW_R	ANGE_VAL, a(SP)+		
				FCE 9	DO D6 51 90	00446	505.	INCL BRW	SUBSCE 16\$	R_COUNT		1118
	1F	A6 08		FCE9	90	004A8 004AB	515:	MOVB CMPZV	CHIDCLE	COUNT, 31 (NODEPTR)	:	1119
A6		08		00 1A	ED 13	004AF 004B5		REOL	524	7. 27 (NODEPTR), SUBSCR_COUNT	•	1119
		3A 7E	00000000	' EF	E8	004B7		BEQL BLBS MOVZBL PUSHL PUSHL CALLS	MULTIF	PLE_SUBSCR, 53\$		1120 1120
		15	18	A6 01	DD	004BE 004C2		PUSHL	#1	DEPTR), -(SP)	•	1120
	00000000	00	00028EA0	8F	DD	00464		PUSHL	#16758	34		
23	0000000G	00 6A		03	FB EO	004CA 004D1	528:	BBS	#19,	IB\$SIGNAL (R10), 53\$		1121

DE

DBGPARSER V04-000					16-Sep-1 14-Sep-1	1984 02:10 1984 12:17):13 :30	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page 354
	02	AA	2/	01	88 00405	BISB2 PUSHL	#1.2	(R10) DEPTR)	: 112
	0000000G	00 5B	24	A6 01	DD 004D9 FB 004DC	CALLS	#1. DE	BG\$STA_TYPEFCODE	: 112
			24	7E A6 5B	D4 004E6 DD 004E8	CALLS MOVL CLRL PUSHL PUSHL MOVQ PUSHL CALLS RET	-(SP)	DEPTR)	112
		7E	04	06	DD 004EB 7D 004ED DD 004F0	MOVQ	FCODE #6	(SP)	1122
	C851	CF	•	AC 06	FB 004F3 04 004F8 53\$:	CALLS	#6, DE	BG\$BUILD_PRIMARY_SUBNODE	1122

; Routine Size: 1273 bytes, Routine Base: DBG\$CODE + 3323

ROUTINE GET_SUBSTRING (PRIMPTR, DSCADDR) : NOVALUE = FUNCTION This routine picks up a substring reference.

for example, in FORTRAN, if a variable X is declared CHARACTER*n then we want to allow X(i:j), where i and j represent the beginning and ending character positions.

In PASCAL, if a variable X is declared PACKED ARRAY[1..N] OF CHAR then we want to allow subscripting X[i] or ranged subscripting X[i:j] to get at substrings of X.

This routine gets called from the GET_SUBSCRIPTS routine, at the point where we discover that what we have is not an array, but is a string.

The expression parser is called to pick up the first subscript. If the terminator ":" is encountered then the expression parser is called again to pick up the upper bound.

These substring bounds then get translated into the PRIM_OFFSET and PRIM_LENGTH fields of the Primary Descriptor, and the SUBREF flag is lit to indicate that a substring selection has taken place.

INPUTS

PRIMPTR - A pointer to the Primary Descriptor for a string about to be subscripted. DSCADDR - A pointer to the string descriptor representing the string to be subscripted.

OUTPUTS The PRIMPTR Primary Descriptor is changed to include the substring information.

BEGIN

PRIMPTR: REF DBG\$PRIMARY DSCADDR: REF DBG\$STG_DESC;

HIGH VALUE, LOW VALUE, NODEPTR: REF DBGSPRIM_NODE, SAVED RADIX, VALPTR: REF DBG\$VALDESC;

Subscript value Subscript value Pointer to Primary Subnode Temporary to save radix Pointer to a Value Descriptor

Temporarily set the radix to decimal. Then call the Expression Parser to pick up the lower string bound.

SAVED_RADIX = .EXPRESSION_RADIX; EXPRESSION_RADIX = DBG\$K_DECIMAL; VALPTR = DBG\$EXPRESSION_PARSER (FALSE, .SUBSCRIPT_TERM_TBL); EXPRESSION_RADIX = .SAVED_RADIX;

```
Check the terminator code. If there was no terminator, (the input line just ended), signal an error. If the terminator was a comma, this is also an error - we only allow the colon or the closing subscript in this case.
IF .TERMINATOR_CODE EQL TOKENSK_TERM_NONE
    SIGNAL (DBG$_MISCLOSUB);
.TERMINATOR_CODE EQL TOKEN$K_TERM_COMMA
THEN
SIGNAL (DBG$ SYNERREXPR, 1, UPLIT BYTE (%ASCIC ','));
CHARPTR = .CHARPTR + .TERMINATOR_LENGTH;
 Convert the subscript value to longword integer.
LOW_VALUE = CONVERT_TO_INTEGER (.VALPTR);
  Check for ":" terminator. This indicates we also have to pick
  up the high value.
IF .TERMINATOR_CODE EQL TOKEN$K_TERM_COLON
     BEGIN
       Pick up another expression for the high value.
     SAVED_RADIX = .EXPRESSION_RADIX;

EXPRESSION_RADIX = DBG$K_DECIMAL;

VALPTR = DBG$EXPRESSION_PARSER (FALSE, .SUBSCRIPT_TERM_TBL);
     EXPRESSION_RADIX = .SAVED_RADIX;
        Check for any of end-of-line, comma, or colon. These are all
        errors here.
     IF .TERMINATOR_CODE EQL TOKEN$K_TERM_NONE
     THEN
     SIGNAL (DBG$_MISCLOSUB);
IF .TERMINATOR_CODE EQL TOKEN$K_TERM_COMMA
     SIGNAL (DBG$ SYNERREXPR, 1, UPLIT BYTE (%ASCIC ','));
IF .TERMINATOR_CODE EQL TOKENSK_TERM_COMMA
     THEN
     SIGNAL (DBG$ SYNERREXPR, 1, UPLIT BYTE (%ASCIC ':'));
CHARPTR = .CHARPTR + .TERMINATOR_LENGTH;
     ! Convert the value descriptor to an integer.
     HIGH_VALUE = CONVERT_TO_INTEGER (.VALPTR);
     END
  No high value present - same as low value.
ELSE
     HIGH_VALUE = .LOW_VALUE;
! Signal an error if the range is reversed.
```

```
DBGPARSER
                                                                                                    16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                          VAX-11 Bliss-32 V4.0-742 [DEBUG.SRCJDBGPARSER.B32;1
                                                                                                                                                                                                   Page 357
(44)
V04-000
                                                        IF .LOW_VALUE LSS 1
OR .HIGH_VALUE GTR .DSCADDR[DSC$W_LENGTH]
OR .LOW_VALUE GTR .HIGH_VALUE
                                                         THEN
                                                              SIGNAL (DBG$_SUBSTRING, 3, LOW_VALUE, .HIGH_VALUE, .DSCADDREDSC$W_LENGTH]);
                                                         ! Signal an error if the values are too large to fit in a Primary.
                                                         IF .LOW_VALUE GTR %X'7FFF'
                                                         THEN
                                                         IF (1+.HIGH_VALUE-.LOW_VALUE);
                                                         THEN
                                                               SIGNAL (DBG$_ILLSUBLEN);
                                                         ! Modify the primary to indicate the substring information
                                                        NODEPTR = .PRIMPTR [DBG$L PRIM BLINK];
PRIMPTR [DBG$V DHDR SUBREF] = TRUE;
PRIMPTR [DBG$W PRIM OFFSET] = .LOW VALUE;
PRIMPTR [DBG$W PRIM LENGTH] = 1 + .HIGH VALUE - .LOW VALUE;
NODEPTR [DBG$L PNODE RELOC] = -1;
                                                        END:
                                                                                                                    .PSECT DBG$PLIT, NOWRT, SHR, PIC, 0
                                                                                              033CF P.AYN:
033D1 P.AYO:
033D3 P.AYP:
                                                                                       01
01
01
                                                                                                                    .ASCII
                                                                                                                                <1>1,1
                                                                                                                    .ASCII
                                                                                                                    .PSECT
                                                                                                                                DBG$CODE, NOWRT, SHR, PIC.O
                                                                                      OOFC 00000 GET_SUBSTRING:
                                                                                                                                Save R2,R3,R4,R5,R6,R7
P.AYN, R7
LIB$SIGNAL, R6
EXPRESSION_RADIX, R5
EXPRESSION_RADIX, SAVED_RADIX
#10, EXPRESSION_RADIX
SUBSCRIPT_TERM_TBL
                                                                                                                                                                                                         1122
                                                                                                                    . WORD
                                                                 000000000°
                                                                                                                    MOVAB
                                                                                         9EE00004B00520B120
                                                                                                                    MOVAB
                                                                                                                    MOVAB
                                                                                                                                                                                                         1127
1127
1128
                                                                                                                    MOVL
                                                                                                                    MOVL
                                                                           20
                                                                                                                    PUSHL
                                                                                                                    CLRL
                                                                                                                                 -(SP)
                                                             CF
54
65
                                                                                                                                #2. DBG$EXPRESSION_PARSER
RO. VALPTR
                                                  CDB5
                                                                                                                    CALLS
                                                                                                                    MOVL
                                                                                                                                SAVED RADIX, EXPRESSION_RADIX TERMINATOR_CODE
                                                                                                                    MOVL
                                                                                                                    TSTL
                                                                           28
                                                                                                                    BNEQ
                                                                  00028E90
                                                                                                                    PUSHL
                                                                                                                                 #167568
                                                                                                                                                                                                         1129
                                                                                                                    CALLS
                                                                                                                                 #1, LIB$SIGNAL
                                                                                                                                 TERMINATOR_CODE, #1
                                                                            28
                                                                                                                                                                                                         1129
                                                                                                                    BNEQ
                                                                                                                    PUSHL
                                                                                                                                                                                                      : 1129
```

DBGPARSER V04-000					1	G 9 6-Sep- 4-Sep-	1984 02:10:1 1984 12:17:3	VAX-11 Bliss-32 V4.0-742 CDEBUG.SRCJDBGPARSER.B32;1	Page 356 (44
			000289E2	01 8F 03	DD 00043 DD 00045 FB 00048		PUSHL A	71 7166370	•
		FBF4	66 20	AS	CO 0004E	28:	CALLS A ADDL2 PUSHL CALLS	73, LIB\$SIGNAL TERMINATOR_LENGTH, CHARPTR	112
		EC26	CF 53 03 28	01	DD 00054 FB 00056 D0 00058		CALLS	VALPTR V1, CONVERT TO_INTEGER	112
			ó3 28	A5	D1 0005E		CMPL 1	RO, LOW VALUE TERMINATOR_CODE, #3	113
			52 65 20	65 0A A5	DD 00045 FB 00046 DD 00056 DD 00056 DD 00056 DD 00056 DD 00067 DD 00067 DD 00077		MOVL CMPL BNEQ MOVL MOVL PUSHL CLRL CALLS	XPRESSION_RADIX, SAVED_RADIX V10, EXPRESSION_RADIX SUBSCRIPT_TERM_TBL -(SP)	113 113 113
		CD68	CF 54 65	02	FB 0006F		CALLS A	2. DRGSEXPRESSION PARSER	
			65 28	52 A5 09	DO 00074 DO 00077 D5 0007A		MOVL TSTL	RO, VALPTR SAVED RADIX, EXPRESSION_RADIX TERMINATOR_CODE	113
			66 00028E90	8F 01	DD 0007F		PUSHL	1167568 11, LIB\$SIGNAL	113
			01 28	A5 OE	D1 00088	3\$:	CMPL	FERMINATOR_CODE, #1	113
			02	01	9F 0008E		PUSHAB PUSHL	AYO	113
			000289E2 66 01 28	8F 03	12 00080 9F 0008E DD 00091 DD 00093 FB 00099		CALLS	1166370 13, LIB\$SIGNAL	1
				A5 OE A7	D1 00090	45:	BNEQ	TERMINATOR_CODE, #1	113
			04 000289E2	01 8F 03	9F 000A2 DD 000A5 DD 000A7 FB 000AD		PUSHL A	P. AYP V1 V166370	113
		FBF4	66 65 20	03 A5	12 000A0 9F 000A2 DD 000A5 DD 000A7 FB 000A0 CO 000B0	58:	CALLS	73. LIB\$SIGNAL TERMINATOR_LENGTH, CHARPTR	113
		EBC4		54	DD 000B6		PUSHL V	ALPTR	113
			CF 52	50	DD 000B6 FB 000B8 D0 000BD		PUSHL V CALLS A MOVL F BRB	11. CONVERT TO_INTEGER RO, HIGH_VACUE	1130
			52	53	DO 000C2	6\$: 7\$:	MOVL L	OW VALUE, HIGH VALUE	113 113 113
52	08	ВС	10	00	15 000C7		CMPZV A	OW_VALUE 00, #16, adscaddr, HIGH_VALUE	113
			52	53	D1 00001		CWAF F	OW_VALUE, HIGH_VALUE	113
			7E 08	BC 52 53	DO 000C2 D5 000C5 15 000C7 ED 000C7 D1 000D1 15 000D4 3C 000D6 DD 000D6 DD 000D6 DD 000D6	8\$:	BLSS CMPL BLEQ MOVZWL PUSHL PUSHL PUSHL CALLS CMPL BLEQ PUSHL PUSHL PUSHL A PUSHL A I RUSHL A I RUSH A	DSCADDR, -(SP) HIGH VALUE LOW_VALUE V3	113
			00028008	03 8F	DD 000DE		PUSHL A	164056	
		00007FFF	66 8F	55	D1 000E9		CMPL	1164056 15. LIB\$SIGNAL OW_VALUE, #32767	113
				53	15 000F0 DD 000F2 DD 000F4		PUSHL L	OW_VALUE	113
			66 000280F0	53 01 8F 03 52	DD 000F4 DD 000F6 FB 000F0 D6 000FF		PUSHL	1164080	
				52	D6 000FF	10\$:	INCL	13. LIB\$SIGNAL	: 113

66 01 FB 00111 CALLS #1, LIB\$SIGNAL 50 04 AC DO 00114 11\$: MOVL PRIMPTR, RO 51 18 AO DO 00118 MOVL 24(RO), NODEPTR	DBGPARSER V04-000			H 9 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 359
		12 A0	50 000280F8 66 50 51 04 51 18 04 10 80 52	8F DD 0010B PUSHL #164088 01 FB 00111 CALLS #1, LIB\$SIGNAL AC DO 00114 11\$: MOVL PRIMPTR, RO AO DO 00118 MOVL 24(RO), NODEPTR 02 88 0011C BISB2 #2, 4(RO) 53 BO 00120 MOVW LOW_VALUE, 16(RO)	1139 1139 1136 1136 1136

ROUTINE OPERATOR_TO_RESTORE_RADIX =

This routine returns the Operator Lexical Token Entry for the operator which will restore the currently set expression radix. It is used in the processing of the lexical operators %DEC, %HEX, %OCT, and %BIN. If the current radix is decimal, for example, when the %HEX operator is encountered, then the decimal radix must be restored when the range of the %HEX operator ends. Consider this example:

10 + %HEX (20 + 30) - 40

Here 10 and 40 are interpreted as decimal numbers while 20 and 30 are treated as hexadecimal numbers. When the %HEX operator is encountered, this routine is called and returns the %DEC operator. The %DEC operator is pushed onto the operator stack and the radix is then set to hexadecimal. When the minus sign is encountered, it forces evaluation of the stacked %DEC operator which restores the radix to decimal.

The current radix value is stored in and picked up from the OWN variable EXPRESSION_RADIX.

INPUTS

NONE

OUTPUTS

The return value of this routine is the address of the Operator Lexical Token Entry for the lexical operator which restores the current value of EXPRESSION_RADIX.

BEGIN

Based on the current radix value, return the lexical operator which will restore that radix value when popped from the operator stack and evaluated.

IF .EXPRESSION_RADIX EQL DBG\$K_DECIMAL THEN RETURN RADIX_OP_DEC:
IF .EXPRESSION_RADIX EQL DBG\$K_HEX THEN RETURN RADIX_OP_HEX:
IF .EXPRESSION_RADIX EQL DBG\$K_OCTAL THEN RETURN RADIX_OP_OCT:
IF .EXPRESSION_RADIX EQL DBG\$K_BINARY THEN RETURN RADIX_OP_BIN;

! Any other current radix value is an internal DEBUG error.

\$DBG_ERROR('DBGPARSER\OPERATOR_TO_RESTORE_RADIX');
RETURN 0;

END:

DBG VO4	PARS	ER													1	9 5-Sep-19 5-Sep-19	984 02:10 984 12:17	0:13 VAX-11 Bliss-32 V4.0-742 Pa 7:30 [DEBUG.SRC]DBGPARSER.B32;1	ige 361 (45)
52 45	45	50 4F	4F 54	5C 53	52	45	53 5F	52 4F	41 54	50 5F	47 52	42 4F 41	44	23	033D5 033E4	P.AYQ:	.ASCII	\#DBGPARSER\<92>\OPERATOR_TO_RESTORE_RAD\	•
											44	41	54	23 41 5F 49	033D5 033E4 033F3 033F7		.ASCII		
																	.PSECT	DBG\$CODE,NOWRT, SHR, PIC.O STORE_RADIX: Save R2 RADIX_OP_DEC, R2 EXPRESSION_RADIX, R1 R1, #10 1\$ RADIX_OP_DEC, R0 R1, #16 2\$ RADIX_OP_HEX, R0 R1, #8 3\$ RADIX_OP_OCT, R0 R1, #2 4\$ RADIX_OP_BIN, R0 P.AYQ	
														0004	00000	OPERATO	OR_TO_RES	STORE_RADIX:	. 1174
										52	00000	000:	EF	9E	00002		MOVAB	RADIX OP DEC. R2	: 1136
										52 51 0A	00000	000	EF 51 04 62	9E 00 01 12	00000		CMPL	R1, #10	1140
										50			62	9E	00015		MOVAB	RADIX_OP_DEC. RO	:
										10			51	04	00018 00019	15:	RET	R1, #16	: 1140
										50		11	51 05 A2	12 9E	0001C 0001E		MOVAB	RADIX OP HEX. RO	
										08				9E4 01 12 9E4 01	00022	28:	RET	R1. #8	1140
										50		22	51 05 A2	12 9F	00026		BNEQ	RADIX OF OCT. BO	
										02				04	00020	74.	RET	D1 #2	1140
										50		33	51 05 A2	12	00030 00032 00036 00037 0003B 0003D	J	BNEQ	4\$ PANTY OD DIN DO	: "
										20				9E 04 9F	00036		RET	RADIA_UP_BIN, RU	1
+												0A9	01	VV	00037 0003B	48:	PUSHAB		1141
							000	0000	OG	00	00028	362	01 8F 03	DD FB	00045		PUSHAB PUSHL PUSHL CALLS	#164706 #3. LIB\$SIGNAL RO	
													50	04	0004A		CLRL	RO	1141

; Routine Size: 77 bytes, Routine Base: DBG\$CODE + 394A

ROUTINE PATHNAME_TO_PRIMARY(PATHDESC, SUBSCR_DESC, PLIPTR, SAVED_PATHDESC, ARRAY_FLAG) =

FUNCTION This routine builds a Primary Descriptor Root Node in temporary memory and returns a pointer to that descriptor. The symbol for which the descriptor is built is identified by a Pathname Descriptor. This routine passes the Pathname Descriptor to the GETSYMBOL routine to get the symbol's SYMID, and then builds the Primary Descriptor from that SYMID. For data items, the symbol's FCODE and TYPEID (which identify the data type) are also retrieved and added to the Primary Descriptor.

After the Root Node has been build, DBG\$BUILD_PRIMARY_SUBNODE is called to build a Primary Descriptor Sub-Node for the symbol. This means that the Primary Descriptor is complete when PATHNAME_TO_PRIMARY returns unless further subscripting or other qualification causes additional Sub-Nodes to be appended later.

INPUTS

PATHDESC - The address of a Pathname Descriptor which defines the symbol for which a Primary Descriptor is to be built.

SUBSCR_DESC - Some languages collect their subscripts as they pick up the Primary, and do not apply them until the end. In this case, SUBSCR_DESC is a data structure containing the saved subscripts.

PLIPTR - In a PL/I exression of the form A->B, we first save away then Primary for "A". This Primary is in PLIPTR, and must be appended to the beginning of the Primary we now build.

SAVED PATHDESC: PTH\$PATHNAME -This is an area of storage in which we can save away a copy of the pathname. This is used to make pathnames sticky in expressions such as P1\A->B

ARRAY_FLAG - An optional fifth parameter, which, if present, indicates that this routine was called as part of an array subscripting operation. This is used in BASIC, where there may be two variables A, one an array and one not, and it is determined from context which is meant. This flag is just passed along to GETSYMBOL so that it can resolve the ambiguity properly.

OUTPUTS

A Primary Descriptor for the symbol specified by PATHDESCR is built and its address is returned as the routine value.

BEGIN

PATHDESC: REF PTH\$PATHNAME, ! Pointer to input Pathname Descriptor SAVED PATHDESC: REF PTH\$PATHNAME, ! Saved copy of pathname. SUBSCR_DESC: REF SUBSCR\$DESC; ! Pointer to subscript information

CH\$MOVE(DBG\$K_PATHNAME_SIZE, .PATHDESC, .SAVED_PATHDESC)

IF .SAVED_PATHDESCEPTH\$B_PATHENT] GTR 1
THEN

:11442

Page 363 (46)

```
DBGPARSER
V04-000
                                                                                                             16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                      VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32:1
11443
                                                                                  BEGIN
                                                                                  LOCAL
:11446
:11447
                                                                                     Merge the previous pathname into this one.
11449
11450
11451
11452
11453
11455
                                                                                  PATHVECTOR1 = PATHDESC[PTH$A PATHVECTOR];
PATHVECTOR2 = SAVED PATHDESC[PTH$A PATHVECTOR];
I = .SAVED PATHDESC[PTH$B PATHCNT]=1;
                                                                                 DECR J FROM .PATHDESCLPTHSB TOTCNTJ-1 TO 0 DO

PATHVECTOR1[.I+.J] = .PATHVECTOR1[.J];

DECR J FROM .I-1 TO 0 DO

PATHVECTOR1[.J] = .PATHVECTOR2[.J];

PATHDESC[PTH$B_TOTCNT] = .PATHDESC[PTH$B_TOTCNT] + .I;

PATHDESC[PTH$B_PATHCNT] = .PATHDESC[PTH$B_PATHCNT] + .I;
                                                                                   ! <<< INVOCNUM
                                                                                  END:
 :11460
                                                                    END
:11461
:11462
                                                             ELSE
11463
11464
11465
                                                                    BEGIN
                                                                       Allocate space for the Primary Descriptor and fill in the descriptor
:11466
                                                                        header fields and sub-node links.
:11467
                                                                   PRIMPTR = DBG$GET TEMPMEM(DBG$K PRIMARY SIZE);
PRIMPTR[DBG$B_DHDR_TYPE] = DBG$K PRIMARY_DESC;
PRIMPTR[DBG$B_DHDR_LANG] = .DBG$GB_LANGUAGE;
PRIMPTR[DBG$W_DHDR_LENGTH] = DBG$K_PRIMARY_SIZE**UPVAL;
PRIMPTR[DBG$L_PRIM_FLINK] = PRIMPTR[DBG$A_PRIM_FLINK];
PRIMPTR[DBG$L_PRIM_BLINK] = PRIMPTR[DBG$A_PRIM_FLINK];
:11468
11469
11470
11471
:11472
:11473
11474
11475
11476
                                                                       Save away copy of first pathname that we see.
                                                                         .PATHDESC[PTH$B_PATHCNT] GTR 1
                                                                           CH$MOVE(DBG$K_PATHNAME_SIZE, .PATHDESC, .SAVED_PATHDESC)
 :11480
 11481
                                                                           SAVED_PATHDESC[PTH$B_PATHCNT] = 0;
                                                                    END:
                                                             DBG$GL_CURRENT_PRIMARY = .PRIMPTR;
 11485
 11486
11487
11488
                                                                Call GETSYMBOL to get the KIND and SYMID for the symbol. We pass along the flag saying whether this symbol was seen in a subscripted form - getsymbol can use that to resolve ambiguities
  11489
                                                                 in BASIC.
 11490
                                                                 This also gives the scope where the symbol
 11491
11492
                                                                 was looked up and we save that in the Primary Root Node.
                                                                 Then case on the KIND to decide what to do next.
 11493
 11494
                                                                  . COMPONENTS_IN_PATHNAME
  11495
   11496
                                                                    ARR_FLAG = .SUBSCR_DESC[O, SUBSCR$B_SUBCNT] GTR O
  11497
                                                                    ARR_FLAG = ACTUALCOUNT() GTR 4;
:11499
                                                             DBG$STA_GETSYMBOL (.PATHDESC, SYMID, KIND, SCOPE_STATE, SCOPE, .ARR_FLAG, FALSE);
```

Page 364 (46)

```
DBGPARSER
V04-000
                                                                                              16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                 VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                                     PRIMPTR[DBG$B_PRIM_SCOPE_STATE] = .SCOPE_STATE;
PRIMPTR[DBG$L_PRIM_SCOPE] = .SCOPE;
                                                     SELECTONE .KIND OF
                                                             Handle the case where the symbol is not found in DEBUG's symbol
                                                              table (i.e., the RST). Signal the appropriate error message.
                                                           [RST$K_INVALID]:
                                                                   First check for Predefined Identifier reserved by the language. Note: We do this after we are sure that the symbol is not in
                                                                   the symbol table. In PASCAL you may redefine the predefined symbols. Currently, PASCAL is the only language where we have predefined identifiers. FORTRAN has them but they are prefaced by a dot "." and are picked up by now. Check that no invocation number is present.
                         1605
                         1606
                         1607
                        1608
                                                                 IF .PATHDESC[PTH$B_LOCINVOC] EQL 0
                        1610
                                                                       THEN
                                                                            BEGIN
                                                                            INCR I FROM O TO .PRIDTBL[-1] - 1 DO BEGIN
                                                                                  PRID = .PRIDTBL[.I] + TABLEBASE;
IF .PRID[PRID$B_KIND] EQL PRID$K_CONSTANT
THEN
                                                                                        BEGIN
                                                                                        LUCAL
                                                                                              TEMP_NAME : REF VECTOR [,BYTE];
                                                                                        THEN
                                                                                              BEGIN
                                                                                              PRIMPTR = CREATE_PRID_CONSTANT(.PRID);
RETURN .PRIMPTR;
                                                                                              END:
                                                                                        END:
                                                                                  END;
                                                                            END:
                                                                     .DBG$GB_LANGUAGE EQL DBG$K_COBOL
                                                                       DBG$NCOB_PATHDESC_TO_CS(.PATHDESC, PATHSTRING)
:11556
                                                                      DBG$NPATHDESC_TO_CS(.PATHDESC, PATHSTRING);
```

Page 365 (46)

Page 367 (46)

Page 368

```
E 10
16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
DBGPARSER
V04-000
                                                                                                                                                                      VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.B32;1
                                                                                                                         IF .SUBSCR_DESC[.SUBSCR_INDEX, SUBSCR$V_ASTER]
                                                                                                                                SIGNAL (DBG$_ILLASTER)
                                                                                                                        ELSE
                                                                                                                                SIGNAL (DBG$_ILLRANGE);
                                                                                                                  IF NOT .NODEPTR[DBG$V_PNARR_RANGE]
                                                                                                                 THEN
                                                                                                                        BEGIN
                                                                                                                        NODEPTREDBG$V_PNARR_RANGE] = TRUE;
INCR K FROM 0 TO .J=1 DO
                                                                                                                                BEGIN
                                                                                                                                SUBVECTOR[.K, DBG$L PNSUB LBOUND] =
.SUBVECTOR[.K, DBG$L PNSUB SVALUE];
SUBVECTOR[.K, DBG$L PNSUB UBOUND] =
.SUBVECTOR[.K, DBG$L PNSUB_SVALUE];
                                                                                                                                END:
                                                                                                                 IF NOT .SUBSCR_DESC[.SUBSCR_INDEX, SUBSCR$V_ASTER]
THEN
                                                                                                                         BEGIN
                                                                                                                            Check for reversed range.
                                                                                                                         IF .SUBSCR_DESC[.SUBSCR_INDEX, SUBSCR$L_LBOUND] GTR
.SUBSCR_DESC[.SUBSCR_INDEX, SUBSCR$L_UBOUND]
                                                                                                                        THEN
                                                                                                                                SIGNAL (DBG$_INVRANSPEC);
                                                                                                                        SUBVECTOR[.J, DBG$L PNSUB_LBOUND] =
.SUBSCR_DESC[.SUBSCR_INDEX, SUBSCR$L_LBOUND];
SUBVECTOR[.J, DBG$L PNSUB_UBOUND] =
.SUBSCR_DESC[.SUBSCR_INDEX, SUBSCR$L_UBOUND];
:11760
:11761
:11764
                                                                                                                 SUBVECTOR[.J, DBG$L_PNSUB_SVALUE] = .SUBVECTOR[.J, DBG$L_PNSUB_LBOUND]
:11766
                                                                                                             fill in the subscript value.
                                                                                                         ELSE
                                                                                                                 BEGIN
                                                                                                                 LOCAL
                                                                                                                         VAL:
                                                                                                                 VAL = .SUBSCR_DESC[.SUBSCR_INDEX, SUBSCR$L_LBOUND];
IF (.VAL LSS .SUBVECTOR[.J. DBG$L_PNSUB_LBOUND]) OR
(.VAL GTR .SUBVECTOR[.J. DBG$L_PNSUB_UBOUND])
:11777
                                11865
:11778
:11779
                               11867
                                                                                                                 THEN
                                                                                                                 SIGNAL (DBG$ SUBOUTBND, 4, .SUBSCR_INDEX, .VAL, .SUBVECTOR[.J, DBG$L_PNSUB_LBOUND], .SUBVECTOR[.J, DBG$L_PNSUB_UBOUND]);
SUBVECTOR[.J, DBG$L_PNSUB_SVALUE] = .SUBSCR_DESC[.SUBSCR_INDEX, SUBSCR$L_LBOUND];
:11780
:11781
:11782
:11784
```

11840

```
F 10
                           16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                    VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.832:1
                    END:
                 If we previously got a range, then make this dimension
                 into a range also.
             IF .NODEPTR[DBG$V_PNARR_RANGE] AND NOT .SUBSCR_DESC[.SUBSCR_INDEX, SUBSCR$V_RANGE]
              THEN
                    BEGIN
                    SUBVECTOR[.J, DBG$L PNSUB LBOUND] =
.SUBVECTOR[.J, DBG$L PNSUB SVALUE];
SUBVECTOR[.J, DBG$L PNSUB UBOUND] =
.SUBVECTOR[.J, DBG$L PNSUB_SVALUE];
              SUBSCR_INDEX = .SUBSCR_INDEX + 1;
          fill in the field that gives the count of subscripts. Also, unless this is an array slice, then light the
          EVAL bit which says that subscripting is being done,
          and tack on a new subnode.
      NODEPTR[DBG$B_PNARR_SUBCNT] = .NODEPTR[DBG$B_PNARR_DIMCNT];
IF NOT .NODEPTR[DBG$V_PNARR_RANGE]
       THEN
             NODEPTR[DBG$V_PNODE_EVAL] = TRUE;
                Attach a new Primary sub-node for the array elements.
             END;
      END:
   If we have just attached a record subnode then fill in the component information here. Note that this must
   be done after the array case above, to properly handle arrays of records.
IF (.FCODE EQL RST$K TYPE RECORD) AND (.COMPONENTS_IN_PATHNAME) AND
     (.1 NEQ 0)
THEN
      BEGIN
         If we are not at the bottom symid, then there must be a symid for the record component below this one. We need to fill in the corresponding record sub-node with the component index. This component index is used by MODIFY PRIMARY to compute logical sucessor/predecessor. We also light the "EVAL" bit in the
```

```
DBGPARSER
V04-000
                                                                                                                                                                                                                                                                16-Sep-1984 02:10:13
14-Sep-1984 12:17:30
                                                                                                                                                                                                                                                                                                                                                               VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32:1
11843
118445
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1184467
1
                                                                                                                                                                                                                        sub-node.
                                                                  11932
11933
11934
11935
11936
11937
11938
                                                                                                                                                                                                                NODEPTR[DBG$V_PNODE_EVAL] = TRUE;
SYMID1 = .SYMID_VECT[.I-1];
TYPCOMPLST = TYPEID[RST$A TYPCOMPLST];
INCR_J_FROM 0 TO .TYPEID[RST$L_TYPCOMPCNT] - 1 DO
                                                                                                                                                                                                                                 SYMID2 = .TYPCOMPLST[.J]:
                                                                 11940
11941
11942
11943
                                                                                                                                                                                                                                       Use the DSTPTR to determine whether we are
                                                                                                                                                                                                                                       really looking at the right component.
                                                                                                                                                                                                                                           .SYMID1[RST$L_DSTPTR] EQL .SYMID2[RST$L_DSTPTR]
                                                                  11944
                                                                                                                                                                                                                                                 BEGIN
                                                                  11946
                                                                                                                                                                                                                                                 NODEPTREDBG$W_PNREC_INDEX] = .J + 1;
                                                                                                                                                                                                                                                 EXITLOOP;
                                                                  11948
                                                                                                                                                                                                                                                 END:
                                                                  11949
                                                                  11950
11951
                                                                                                                                                                                                                                       We should not fall through to here.
                                                                                                                                                                                                                                 END:
                                                                  11952
                                                                                                                                                                                                                END:
                                                                                                                                                                                                 END:
                                                                  11954
11955
                                                                                                                                                                                        If we have not exhausted the given list of subscripts, then first check for a substring reference.
                                                                  11956
11957
                                                                                                                                                                                       We must have seen something that looks like a ranged subscript "(i:j)". In order for a substring to be legal then the data type must
                                                                  11958
11959
                                                                 11960
11961
11962
11963
11964
11965
11966
11967
11968
11969
11970
                                                                                                                                                                                        either be text or one of the decimal string types.
                                                                                                                                                                               PICKED_UP_SUBSTRING = 0;
IF (.SUBSCR_INDEX EQL (.SUBSCR_DESC[0, SUBSCR$B_SUBCNT]-1))
AND .SUBSCR_DESC[.SUBSCR_INDEX, SUBSCR$V_RANGE]
AND .SUBSCR_DESC[.SUBSCR_INDEX, SUBSCR$V_MARKER]
AND (NOT .SUBSCR_DESC[.SUBSCR_INDEX, SUBSCR$V_ASTER])
                                                                                                                                                                                 THEN
                                                                                                                                                                                                DTYPE = DSC$K DTYPE Z:
IF .KIND EQL RST$K DATA
                                                                  11971
                                                                                                                                                                                                 THEN
                                                                 11972
11973
11974
                                                                                                                                                                                                                 SELECTONE . FCODE OF
                                                                                                                                                                                                                             CRSTSK_TYPE_ATOMIC]:
BEGIN
TYP ATOM
                                                                  11975
                                                                  11976
                                                                                                                                                                                                                                                DBG$STA_TYP_ATOMIC(.TYPEID, DTYPE, BITSIZE);
LEN = .BITSIZE / 8;
                                                                  11978
11979
                                                                                                                                                                                                                               [RSTSK_TYPE_DESCR]:
                                                                 11980
11981
11982
11983
11984
11985
                                                                                                                                                                                                                                                DBG$STA_TYP_DESCR(.TYPEID, DESCR);
DTYPE = .DESCR[DSC$B_DTYPE];
11895
11896
11897
11898
                                                                                                                                                                                                                                                 LEN = .DESCR[DSC$W_LENGTH];
                                                                  11986
                                                                                                                                                                                                                                [RST$K_TYPE_PICT, RST$K_TYPE_RECORD]:
```

END

ELSE

1195

We have seen something that looks like a substring but the data type is wrong. Catch that case here and signal an error.

```
DBGPARSER
V04-000
                                                                                                     VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                                                SIGNAL (DBG$_ILLSUBSTR);
                                                            END
                                                       ELSE
                                                              We have seen something that looks like a substring but the data type is wrong. Catch that case here and
                                                              signal an error.
                                                            SIGNAL (DBG$_ILLSUBSTR);
                                                       END:
                                                    Signal an error if we picked up too many or too few subscripts.
                                                      .TOOFEWSUB
                                                      SIGNAL (DBG$ TOOFEWSUB, 1, .EXPECTED_SUBS);
.SUBSCR_INDEX LSS .SUBSCR_DESC[0, SUBSCR$B_SUBCNT]
                                                       SIGNAL (DBG$_TOOMANSUB, 1, .SUBSCR_INDEX-.PICKED_UP_SUBSTRING);
                                                  END:
                                                Anything else we treat as an internal DEBUG error.
                                              [OTHERWISE]:
                                                  $DBG_ERROR('DBGPARSER\PATHNAME_TO_PRIMARY 10');
                                              TES:
                                           Return a pointer to the Primary Descriptor to the caller.
                                         RETURN .PRIMPTR;
                                         END:
                                                                                     .PSECT DBG$PLIT,NOWRT, SHR, PIC,0
                                                                     033F9 P.AYR:
                                                                                     .ASCII \2DBGPARSER\<92>\PATHNAME_TO_PRIMARY sym\
                                                                                     .ASCII \id stack overflow\
                                                                            P.AYS:
                                                                                     .ASCII \ DBGPARSER\<92>\PATHNAME_TO_PRIMARY 10\
                                                                                     .PSECT DBG$CODE, NOWRT, SHR, PIC, O
                                                               Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
-284(SP), SP
PATHDESC, R8
                                                                                                                                                 : 1141
                                             5E
                                                                                                                                                 : 1152
                                                                                     MOVL
```

BGPARSER 104-000								J 10 16-Sep 14-Sep	-1984 02:10 -1984 12:17	:13	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page 376
				59	10 04 00	AC AE AC SO	D0 D4 D5	0000B 0000F 00012	MOVL CLRL TSTL BEQL	SAVE 4(SP PLIP 5\$	D_PATHDESC, R9	115
			0A	5B 56 A6 01	04 00 18 01	ACC	D6 D0 D0 88	00017 0001A 0001E 00022 00026	BEQL INCL MOVL BISB2 CMPB BGTRU CMPB BLEQU MOVAB MOVAB MOVAB MOVZBL DECL MOVZBL	1. (CD	PTR, PRIMPTR PRIMPTR), NODEPTR 10(NODEPTR) 3), #1	115 115 115 115
				01	01	67 A9	1A 91	00034	BGTRU CMPB	1(R9	0, #1	115
				57 52 51	08 08 01	A8 A9 A9	18 9E 9A	0002A 0002C 00030 00032 0003A 0003A 00040 00043 00045 1\$:	MOVAB MOVAB MOVZBL	8\$ 8(R8 8(R9 1(R9), PATHVECTOR1), PATHVECTOR2), I	115 115 115
				50		68 09 50	94	00040	MOVŽBL BRB ADDL3	(R8)		115
		53	6	743 F4 50	•	6740 50 51 05	DO F4	00047	MOVL	(PAT J. 1	HVECTOR1)[J], (PATHVECTOR1)[R3]	115
			01	740 F8 68 A8		6240 50 51	P 6 8 6 8 6 8 6 8 6 8 6 8 6 8 6 8 6 8 6	0005B 4\$:	MOVL BRB MOVL SOBGEQ ADDB2 ADDB2 BRB PUSHL CALLS	1. (HVECTOR2)[J], (PATHVECTOR1)[J] (R8)	115 115 115 115
			000000006	00		35 09 01 50	DD FB	00065 00067 00069	BRB PUSHL CALLS	#0	DBG\$GET_TEMPMEM	115
			02 03	5B AB AB 6B	00000000G	8F	90 90 80	00070 00073 00078 00080	MOVE MOVB MOVW	#121 DBG\$	DBG\$GET_TEMPMEM PRIMPTR , 2(PRIMPTR) GB_LANGUAGE, 3(PRIMPTR) (PRIMPTR)	115 115
			14 18	AB AB 01	14 14 01	AB AB AB 06 34	9E 9E 91	00083 00088 0008D	MOVAB MOVAB CMPB	20(P 20(P 1(R8	GÉB LANGUAGE, 3(PRIMPTR) (PRIMPTR) (PRIMPTR), 20(PRIMPTR) (PRIMPTR), 24(PRIMPTR) (S), #1	115 115 115 115 115 115
		69		68		34	1B 28	00091	MOVC3	#52.	(R8), (69)	1150
			000000006	00	01 00000000°	03 A9 5B EF 50	94 D0 E9	00099 7\$: 00090 8\$: 0000A3	MOVE MOVB MOVB MOVAB MOVAB CMPB BLEQU MOVC3 BRB CLRB MOVL BLBC CMPZV BGTR BRB CLRL CMPB BLEQU INCL	1(R9 PRIM COMP	PONENTS_IN_PATHNAME, 9\$	1156 115 1158
00	08	BC		08		08 09 09	ED	000AC 000B2	CMPZV BGTR	110	#8, asubscr_desc, #0	
				04		50 60 02	11 04 91	000B4 000B6 9\$: 000B8	BRB CLRL CMPB BLEQU	11\$ R0 (AP) 11\$. #4	1158
				51		50 7E 51	D6	000BD 10\$: 000BF 11\$: 000C2	INCL MOVL CLRL PUSHL PUSHAB PUSHAB PUSHAB	RO RO	ARR_FLAG	1158
					20 34 30	AE AE	9F 9F	00000	PUSHAB PUSHAB	SCOP	FLAG E_STATE	

51

					1	K 10 6-Sep-1 4-Sep-1	984 02:10 984 12:17	:13 VAX-11 Bliss-32 V4.0-742 Page 30 EDEBUG.SRCJDBGPARSER.B32;1	ge 375 (46)
			44 AI	9F	000CF 000D2		PUSHAB	SYMID	
	0000000gG	00	28 AI 24 AI 20 AI	DD B 900 0315 9100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	000D2 000D4		PUSHL CALLS MOVB MOVL MOVL BEQL BRW	R8 #7, DBG\$STA_GETSYMBOL	
	10	AB AB 53	28 AI 24 AI 20 AI	90 00	00004 0000B 000E0 000E5		MOVB	#7, DBG\$STA_GETSYMBOL SCOPE_STATE, 28(PRIMPTR) SCOPE, 32(PRIMPTR) KIND, R3	1158
		53	SC VI	D0	000E5 000E9		MOVL	KIND, R3	1159
			008 02 A	31	OOOEB	12\$:	BRW	12\$ 18\$ 2(R8) 15\$	1160
		50	02 A	12	000F1	120.	BNEQ	15\$ PD 10 TP1 PO	:
		50 59 55	FC A	00	000FA		MOVL	PRIDTBL, RO -4(RO), R9	1161
			9		000EB 000F1 000F3 000FA 000FB 00103 00116 00116 00116 00117 00126 00126 00130 00137		MNEGL BRB	14\$ I	
54		50	00000000 FF4	9E	00103 0010A	13\$:	MOVAB ADDL3	TABLEBASE, RO APRIDTBL[I], RO, PRID (PRID), #1	1161
		01	6	91	00113		CMPB BNEQ		1161
		57 50 52 51	08 Ā	9E C1 91 12 9E 00 9A	00118		CMPB BNEQ MOVAB MOVL MOVZBL MOVZBL	8(R8), PATHVECTOR1	1162
		52	08 A	9A	0011F		MOVZBL	8(PRID) R2	1162
00	09	Ã4	5	9A 2D	00126		CMPC5	8(R8), PATHVECTOR1 (PATHVECTOR1), TEMP_NAME 8(PRID), R2 (TEMP_NAME), R1 R2, 9(PRID), #0, R1, 1(TEMP_NAME)	1162 1162 1162
			08 A	12	0012E		BNEQ	14\$	
	EA65	CF 5B	0	FB	00130		PUSHL	PRID #1, CREATE_PRID_CONSTANT RO, PRIMPTR	1163
			050) DO	00137 0013A		BRW	RO, PRIMPTR	1163
CS		55	0500 00000000G 00 34 AE	12 DD FB DD 31 F2 91 29 91	0012C 0012E 00130 0013Z 00137 0013A 0014B	14\$: 15\$:	AOBLSS	PO 1 13¢	1161
			34 A	12	00148 0014A		CMPB BNEQ PUSHAB	DBG\$GB_LANGUAGE, #3 16\$ PATHSTRING	1164
	00000000	00	54 58	DD	0014D		PUSHL	R8	1104
	000000006	00		11	00156		CALLS BRB PUSHAB	#2 DBG\$NCOB_PATHDESC_TO_CS	
			34 AE	FB 11 9F DD DD DD DD 11	00158 0015B	16\$:	PUSHAB	PATHSTRING R8	1164
	0000000G	00	34 A	FB	0015D 00164	175:	PUSHL	R8 #2. DBG\$NPATHDESC_TO_CS PATHSTRING	1164
			000281F8 8F	DD	00167		PUSHL CALLS PUSHL PUSHL PUSHL	#1 #164344	
		09	34 AE 07 34 AE 07 000281F8 8F	11	0016F 00171	18\$:	BRB CMPL BEQL CMPL BNEQ CMPB BNEQ PUSHAB	23\$ R3, #9 19\$	1165
		OD	Q:	D1 13	00174		BEQL	19\$ R3, #13	
			000000006 00	12	00179	198:	BNEQ	24\$	1165
		03	7, 0	12	00182	170.	BNEQ	DBG\$GB_LANGUAGE, #3	
	00000000		34 AI	DD	00187		PUSHAB	PATHSTRING R8	1165
	0000000G	00	_ 0	11	00189		PUSHL CALLS BRB PUSHAB PUSHL CALLS	R8 #2. DBG\$NCOB_PATHDESC_TO_CS 21\$	
			34 AI	9F DD	00192	20\$:	PUSHAB	PATHSTRING	1166
	000000006	00	00000000G 00 34 Al	01 12 91 12 9F 0D FB 11 9F 0D FB	0014F 00158 0015B 0015D 00167 00167 0016F 00176 00178 00178 00187 00189 00192 00195 00197	21\$:	CALLS	R8 #2, DBG\$NPATHDESC_TO_CS R3, #9	1166

				1	L 10 6-Sep- 4-Sep-	1984 02:10 1984 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 CDEBUG.SRCJDBGPARSER.B32;1	Page 376
		34	0D AE 01 86 0B AE 01 85 04 85 04 85	12 001A1 DD 001A3 DD 001A6 DD 001A8 11 001AE DD 001B3 DD 001B3 DD 001B5 31 001BB D1 001BE 19 001C1		BNEQ PUSHL PUSHL BRB PUSHL PUSHL PUSHL BMPQ BNEQ BNEQ CLRL PUSHL BRW CLRS CLRS CMPQ CMPQ CLRS CMPQ CMPQ CLRS CMPQ CMPQ CMPQ CMPQ CMPQ CMPQ CMPQ CMPQ	22S PATHSTRING	: 1166
		000281F0	01 8F	DD 001A6		PUSHL	#164336	
		34	OB	11 001AE DD 001B0	225:	BRB PUSHL	PATHSTRING	1166
		000282A8	01 8F	DD 001B3		PUSHL		
	02			31 001BB	23 \$:	CMPL	83\$ R3, #2	1167
	05		53	D1 001C3		CMPL	R3, #5	1
	08		53	D1 001C8	25\$:	CMPL	R3, #8	
07 00	AB	30	53	90 001CD	26\$:	MOVB	#164520 83\$ R3, #2 25\$ R3, #8 27\$ R3, 7(PRIMPTR) SYMID, 12(PRIMPTR) TYPEID -(SP) TYPEID FCODE SYMID R3 PRIMPTR	1168
oc .	ND	30 38	ĄĘ	7C 001D6		CLRO	SYMID, 12(PRIMPTR) TYPEID	1168 1168 1168
		3C 44 3C	AE	DD 001DB		PUSHL	TYPEID	. 1100
		36	AE 53	DD 001E1		PUSHL	SYMID	
C4E8	CF		050525AA7AAA55053501EE15	DD 001A6 DD 001A8 DD 001B3 DD 001B3 DD 001B3 DD 001B3 DD 001B3 DD 001C3 DD 001C3 DD 001C3 DD 001C8 DD 001C8 DD 001C8 DD 001C8 DD 001B4 DD 001B4 DD 001B5 DD 001B6 DD 001B6 DD 001B6 DD 001B6 DD 001B6 DD 001F6 DD 001C3 DD 001F6 DD 001F6 DD 001B6 DD 001F6 DD		PUSHL	PRIMPTR #6, DBG\$BUILD_PRIMARY_SUBNODE	
	06	(045A 53	31 001ED D1 001F0	278:	BRW	#6, DBG\$BUILD_PRIMARY_SUBNODE 84\$ R3, #6 28\$ 82\$ #1, INDEX	1159
		(03	13 001F3 31 001F5		BEQL	28\$ 82\$	
	55 52 06	30 20	O1 AE	CE 001F8 DO 001FB	28\$:	MNEGL	CAMILL DY	1170
	06	50	AE 31	D1 001FF 12 00203	29\$:	MOVL CMPL BNEQ	KIND, #6	1170
	34			D6 00205 D1 00207		TMFI	KIND, #6 31\$ INDEX INDEX, #52 30\$ P. AYR #1	1170
		00000000	EF	19 0020A 9F 0020C		PUSHAB	30\$ P.AYR	1171
00000000	00	00028362	8F	DD 00212 DD 00214		PUSHL	#164706	
00000000G 4C A	00 E45	10	52	D1 00207 19 0020A 9F 0020C DD 00212 DD 00214 FB 0021A D0 00226 D0 00226 D0 00228 9A 0022F 11 00234	30\$:	CMPL BLSS PUSHAB PUSHL PUSHL CALLS MOVL MOVL MOVL MOVZBL	#164706 #3, LIB\$SIGNAL R2, SYMID VECT[INDEX] 16(R2), SYMID SYMID, R2 20(R2), KIND 29\$	1171
	AE 52 AE	10 30 14	AE	DO 0022B		MOVL	SYMID, R2	1171
50	AE	14	69	11 00234	315:	BRB	29\$	1170
	53	1C 01	AE	7C 00238	318:	BRB CLRL CLRQ MOVAB	SUBSCR_INDEX TOOFEWSUB 1(R5), I 64\$	1170 1171 1172 1174
30	AE	40	55 15 16 16 16 16 16 16 16 16 16 16 16 16 16	31 0023F	325:	HDU	64\$ SYMID_VECT[1], SYMID	:
30	55	40 /	59	D4 00248	320.	CLRL	R9	1172
	,,		ÓC 59	12 00240		BNEQ	33\$ R9	
		00	59 53 00 59 AC 05 AE	01 00207 19 0020A 9F 00212 DD 00214 FB 0021A DO 0022B 9A 0022B 9A 0022B 9A 0023B 7C 0023B 7C 0023B 7C 0024A DO 0024A DO 0024A DO 0024A DO 00256 DO 00256		MOVL CLRL CMPL BNEQ INCL TSTL BNEQ MOVL	PLIPTR 33\$	
00	AB	30	ĀĒ	00 00256		MOVL	SYMID, 12(PRIMPTR)	: 1173

DBGPARSER //04-000					M 10 16-Sep- 14-Sep-	1984 02:10: 1984 12:17:	13	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page 37 (46
SC	57 AE	30 14	AE A7 57					D R7 75, KIND	; 117
000000006			57	DO 0025 9A 0025 DD 0026 FB 0026 9F 0026		PUSHL	R7	DRGSSTA SETCONTEXT	117
	•••	38 40	AE AE 57	91 0027	3	MOVIBL PUSHLS PUSHAB PUSHAB PUSHLS PUSHL PUSH	TYPE	DBG\$STA_SETCONTEXT	117
0000000G	00			DD 0027 FB 0027		PUSHL	R7	DBG\$STA_SYMTYPE	
		3¢	03 7E AE AE 57	D4 0027 DD 0027 DD 0028		PUSHL	TYPE	DBG\$STA_SYMTYPE	117
		30		DD 0027 DD 0028 DD 0028 DD 0028 DD 0028 FB 0028		PUSHL	R7 KIND		
C445	CF		AE 5B 06 AB 59	DD 0028 FB 0028	3	PUSHL	PRIM	PTR DBG\$BUILD_PRIMARY_SUBNODE	
	13	18	AB 59	EA 005A	•	MOVL BLBC	24 (P	RIMPTR), NODEPTR 35\$	117
	07	04 30	AE AE O5	E9 0029 D1 0029 12 0029	3	CMPL	FCOD	PTR DBG\$BUILD PRIMARY_SUBNODE RIMPTR), NODEPTR 35\$), 35\$ E, #7	117
		10	A6	04 002A		CLRL	IDIN	UVEFIRA	117
OA	A6 01	30	20 AE 03	88 002A	345:	BISB2 CMPL	#32, FCOD	10(NODEPTR) E, #1	117
	E4	00000000	03 01BB EF	13 002A 31 002B E9 002B 9E 002B 9A 002B CO 002C	36\$: 37\$:	BEQL BRW	A 1 E		•
	F6	28	46	E9 002B 9E 002B 9A 002B))():	MOVAB MOV7BI	40 (R	ONENTS IN PATHNAME, 36\$ (6), SUBVECTOR (DDEPTR), RO EXPECTED_SUBS (DDEPTR), 24(SP) CR_DESC, R9	117
20 18	AE AE 59 57	1B 08	A6 50 A6 AC 01 0158 010	CO 002C	5	ADDL2 MOVZBL	RO. 27(N	EXPECTED_SUBS ODEPTR), 24(SP)	
	57	08	AC 01	9A 002C D0 002C CE 002C 31 002D ED 002D 14 002D 95 002D		MOVL	SUBS	CR_DESC. R9	117 117 118
54 01 A9	08		00	ED 0020	38\$:	CMPZV	#1, 58\$ #0, 40\$ 1(R9	#8, 1(R9), SUBSCR_INDEX	117
		01	A9 04 53	95 002D 12 002E		TSTB BNEQ	1(R9)	117
			53	05 002E		TSTL BEQL			117
10	AE	01 01	A9	94 002E 95 002E 12 002F 9A 002F	39\$:	CLRB	1(R9	TOOFEWSUB) ODEPTR), R1	117 117
	51	18	A9 A9 23 A6 01	12 002F	3	BNEQ MOVZBL	43\$ 27(N	ODEPTR), R1	117
	50		01	CE 002F	3	MNE GL BRB	"1"		
5A	54 5A AA49		0C	C1 002F	41\$:	ADDL3 MULL2	#12.	UBSCR_INDEX, R10 R10	117
02 /		00020000	01 6A49 8F 51	88 0030 9F 0030 C8 0030		BRW CMPZV BGTR TSTB BNEQ TSTL BEQL MOVL CLRB TSTB BNEQ MOVZBL MNEGL BRB ADDL3 ADDL3 ADDL3 ADDL3 ADDL3 ADDL3 ADDL3	(R10	UBSCR_INDEX, R10 R10 2(R10)[R9])[R9] 072, a(SP)+ J, 41\$ J, 8(SP) 8(SP), R0 SUBVECTOR, 12(SP) SUBSCR_INDEX, R10	117
08 AE 50 08 0C AE 5A	9E 50 57	2002000	14	F5 0051	425:	AOBLEQ MULL3	R1 #26.	J. 41\$ J. 8(SP)	117 118 118
0C AE 08	AE 52 54		08 50 00	C5 0031 C1 0031 C1 0031 C5 0032		ADDL3	#8. RO	8(SP), RO SURVECTOR 12(SP)	118

				N 10	4 02:10:1	3 VAX-11 RI iss-32 V4 0-742	Page 378
				16-Sep-198 14-Sep-198	4 12:17:3	VAX-11 Bliss-32 V4.0-742 DEBUG.SRCJDBGPARSER.B32;1	(46)
03	02 AA49		00 E0 00 0090 31 00	0328 032E	BBS #	0, 2(R10)[R9], 44\$	
				0331 448:	TSTL I		1181
		01		0333 0335	BBS # 5 TSTL I BEQL 4 TSTB 1	7\$ (R9)	1181
08	6A49		11 E1 00	033A	BBC #	17. (R10)[R9], 45\$	1181
		00028F68	8F DD 00 06 11 00 8F DD 00 01 FB 00 03 E0 00 08 88 00 01 CE 00 18 11 00 14 C5 00	033F 0345	PUSHL #	167784	1181
	000000006 00	00028F60	8F DD 00	0347 45\$: 034D 46\$: 0354 47\$:	BRB 40 PUSHL # CALLS #	6\$ 167776 1, LIB\$SIGNAL	1182
25	00000000G 00 0A A6 0A A6 50		8F DD 00 01 FB 00 03 E0 00 08 88 00 01 CE 00 18 11 00	0354 47\$: 0359	CALLS # BBS # BISB2 # MNEGL #	3, 10(NODEPTR), 50\$ 8, 10(NODEPTR) 1, K	1182 1182 1182
	50		01 CE 00	0350	MNEGL #	1, K	1182
51	50	00	14 05 00	0360 0362 48\$:	BRB 44 MULL3 #4 PUSHAB 8 PUSHAB (1) MOVL 20 PUSHAB 1 PUSHAB (1)	20, K, R1	1107
		08	A142 9F 00	0366 036A	PUSHAB ((R1)[SUBVECTOR] R1)[SUBVECTOR]	1183
	9E	00	9E DO 00 A142 9F 00 6142 9F 00	036D 0370	PUSHAB 1	R1)[SUBVECTOR] (SP)+, a(SP)+ 2(R1)[SUBVECTOR]	1183
	9E		6142 9F 00	0374 0377	PUSHAB (I	DITELLEVECTORI	
E4	9E 50 6A49		57 F2 00	037A 49\$: 037E 50\$:	MOVL a	(SP)+, a(SP)+ K, 48\$ 17, (R10)[R9], 52\$ (R10)[R9] (R10)[R9] (SP)+, a(SP)+	1182
	0	08 04	AA49 9F 00	0383	BBS # PUSHAB 8 PUSHAB 4	(R10)[R9]	1184
	9E	04	9E D1 00	0388	CMPL a	(SP)+, a(SP)+	
		00028F08	9E D1 00 0D 15 00 8F DD 00 01 FB 00	038E 0390	CMPL a BLEQ 5	67688	1184
50	00000000G 00 57		01 FB 00	0390 0396 0390 51\$:	CALLS # MULL3 # PUSHAB 8 PUSHAB 4 MOVL 9 PUSHAB 1 PUSHAB 8	1, LIB\$SIGNAL 20, J, R0 (R0)[SUBVECTOR] (R10)[R9] (SP)+, a(SP)+ 2(R0)[SUBVECTOR] (R10)[R9] (SP)+, a(SP)+ 8(SP)[SUBVECTOR] 16(SP), a(SP)+ 6\$ (R10)[R9] (SP)+, VAL	1184
		08 04	A042 9F 00	03A1 03A5	PUSHAB 8 PUSHAB 4	(RO)[SUBVECTOR] (R10)[R9]	1184
	9E	00	9E DO 00 A042 9F 00 AA49 9F 00	03A9 03AC	MOVL a	(SP)+, a(SP)+ 2(RO)[SUBVECTOR]	1185
	9E	08	AA49 9F 00	03B0 03B4	PUSHAB 8	(R10)[R9] (SP)+ a(SP)+	
	9E	08	9E 00 00 BE42 9F 00 BE 00 00	03B7 52\$:	MOVL a	8(SP) [SUBVECTOR]	1185
	71.		BE DO 00	03A9 03AC 03B0 03B4 03B7 03BB 03BF 03C1 03C5 03C5 03C6	MOVL a	6\$	1185
	OC BE	04	9E DO 00 50 D1 00	03C1 53\$: 03C5	PUSHAB 4 MOVL a CMPL V	(SP)+, VAL	1186
			0F 19 00	03C8 03CC	BLSS_ 5	(RTO) LR9] (SP)+, VAL AL, a12(SP) 4\$ 12, 8(SP), R1 1, SUBVECTOR, (SP) AL, a0(SP) 5\$ 12, 8(SP), -(SP) (SP)+[SUBVECTOR] (SP)+ 16(SP) AL	1186
51 6E	08 AE 52 00 BE		0C C1 00	03CE 03D3	ADDL3 R	12, 8(SP), R1 1, SUBVECTOR, (SP)	1186
	00 BE		50 D1 00 20 15 00	03D7 03DB	CMPL VI	AL, a0(SP)	
7E	08 AE		0C C1 00	03DD 54\$:	ADDL3 #	12, 8(SP), -(SP)	1187
		10	9E DD O	03E5	PUSHL a	(SP)+	1196
		10	9E DO 00 50 D1 00 51 C1 00 51 C1 00 520 D1 00 20 C1 00 9E42 9F 00 9E42 9F 00 9E42 DD 00 54 DD 00 54 DD 00 54 DD 00 56 FB 00	03E7 03EA 03EC 03EE 03F0 03F6			1186
			04 DD 00	03EE	PUSHL #	UBSCR_INDEX	
	000000006 00	0002868B	8F DD 00	03F6	I OOIIE W	165515 6, LIB\$SIGNAL	

							16 14	11 -Sep- -Sep-	-1984 02:10 -1984 12:17	:13	VAX-11 Bliss-32 V4.0-742 EDEBUG.SRCJDBGPARSER.B32;1	Page 379 (46)
				08	BE42 AA49	9F 0	03FD	55\$:	PUSHAB PUSHAB	38(SP	[SUBVECTOR]	: 1187
	16	04	9E	04	03 00	00 0	03FD 0401 0405 0408	56\$:	MOVL	a(SP))[SUBVECTOR])[R9] +, a(SP)+ 0(NODEPTR), 57\$	1187
	1E 18	02 A	A49	08	00 BE42	EO O	040D 0413	,,,,	MOVL BBC BBS PUSHAB	#0. 2 a8(SP	(R10)[R9], 57\$: 1187 : 1188
	50	0C 08	BE AE 52		9E 0C 50	DO 0	0417 0418		MOVL ADDL3 ADDL3	a(SP)	+, a12(SP) 8(SP), R0	1188
	51			08	BE42	C1 0	0420		PUSHAB	80. S	UBVECTOR, R1)[SUBVECTOR]	
	•••		61		9E	DO 000000000000000000000000000000000000	0428 042B	57\$: 58\$:	MOVL INCL AOBLSS	SUBSC	(R10)[R9], 57\$)[SUBVECTOR] +, a12(SP) 8(SP), R0 UBVECTOR, R1)[SUBVECTOR] +, (R1) R_INDEX), J, 59\$	1188 1176
	02		57	18	AE 03	11 0	042D 0432		BRB	24(SP 60\$ 38\$), J, 59\$; 1176
	20	1F	A6	18	FE9E A6 03	31 00 00 00 00 00 00 00 00 00 00 00 00 00	0428 0428 0420 0432 0434 0437	59 \$:	BRW MOVB	27 (NO	DEPTR), 31 (NODEPTR)	1189
	20	0A 0A 38	A6 A6 AE	24	01	88 0	0441		BBS BISB2 MOVL	#1. 1	O(NODEPTR), 61\$ O(NODEPTR) DEPTR), TYPEID	: 1189 : 1190 : 1190
		00000000		24 38	A6 AE 01	DD C	044A		PUSHL	TYPEI	BG\$STA_TYPEFCODE	1190
		30	00 AE		50	DO 0	0454		MOVL CLRL PUSHL PUSHL MOVQ	RO, F -(SP) TYPEI	CODE	1190
				3C	7E AE AE	DD C	045A		PUSHL	FCODE		: 1190
			7E		06	7D 0	0460		PUSHL	M6	(SP) TR	1190
		C26B	CF 56 07	18 30	5B 06 AB 42 EF 37	DO 0	0465 046A		MOVL	24 (PR	BG\$BUILD PRIMARY_SUBNODE IMPTR), NODEPTR	1190
			3B	00000000	42	12 0	046E 0472 0474	013:	MOVL CMPL BNEQ BLBC TSTL	FCODE 64\$		1191
			36	00000000	53	D5 0	047B 047D		TSTL	1 64\$	NENTS_IN_PATHNAME, 64\$	1191
		0A 14	A6	48	01 AE43	88 0	047F 0483		BISB2 MOVL	WI. 1	O(NODEPTR)	1193 1193
			A6 AE 51 58 50	48 38 20	AE A1	88 0 00 0 00 0 9E 0	0489 048D		MOVL MOVAB MNEGL	TYPE I	O(NODEPTR) VECT-4[I], SYMID1 D, R1), TYPCOMPLST	1193
					01 18	CE 0	0491		MNEGL BRB	#1, J		1194
	57 59	10 10 14	AE AE 67		6840 0C 0C 69	DO 0	047F 0483 0489 048D 0491 0494 0496 04A5 04A8	62\$:	BRB MOVL ADDL3 ADDL3 CMPL BNEQ ADDW3	#12.	OMPLST)[J], SYMID2 SYMID2, R7 SYMID1, R9 (R7)	: 1193 : 1194
	59	14	AE 67		0C 69	DO 0 C1 0 C1 0 D1 0 A1 0	04A0 04A5		ADDL3 CMPL	(R9),	SYMID1, R9 (R7)	
18	A6		50		01	A1 0	04AA		ADDW3	#1. J	. 24(NODEPTR)	1125
	EO		50 02	28	05 A1 53 03	F2 0	04AF 04B1 04B6 04B9 04BB 04C0 04C4	635:	AOBLSS SOBGEQ BRB	70101	, , , , ,	1194 1193 1172
			02		03	11 0	0489	658.	BRB BRW	1, 65 66\$ 32\$		1112
			57	08	FD84	31 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	04BE	66\$:	CLRL	PICKE	D_UP_SUBSTRING R_DESC, R7 , R0	1196 1196
			57	08 01	AC A7 50 54 03	UI U	UNLO		MOVL MOVZBL DECL	1(R7) R0	, RÔ	
			50		03	D1 0	04CA 04CD		CMPL BEQL		R_INDEX, RO	

					16-5 14-5	11 Sep-1984 02:10 Sep-1984 12:17	:13 VAX-11 Bliss-32 V4.0-742 :30 [DEBUG.SRC]DBGPARSER.B32;1	Page 380 (46)
5A		54		0137 00 00 00 12 00	31 004CF 67 C5 004D2 68 E1 004D6	7\$: BRW 8\$: MULL3	80\$ #12, SUBSCR_INDEX, R10	1196
F3	02 A	A47		00	E1 00406 C5 0040C	BBC MULL3	#0. 2(R10)[R7]. 67\$	1196
5A 5A EA 5A	6	A47		12	31 004CF 67 C5 004D2 68 E1 004D6 C5 004E0 C5 004E5 E0 004E9 D4 004EE D1 004F1 12 004F5	BBC MULL3 BBC MULL3	#12, SUBSCR INDEX, R10 #18, (R10)[R7], 67\$ #12, SUBSCR INDEX, R10 #17, (R10)[R7], 67\$	1196
E1	6	A47	44	11 AE	E0 004E9 D4 004EE	BBS	#17, (R10)[R7], 67\$ DTYPE	1196
		06	20	AE 78	D1 004F1 12 004F5	CMPL BNEQ	KIND, #6	1197
		50	30	AE 78 AE 50	DO 004F7	MOVL	FCODE, RO	1197
			40	17 AE	12 004FE	BNEQ PUSHAB	RO, #2 69\$ BITSIZE	1197
			40 48 40	AE AE O3	9F 00503 DD 00506	PUSHAB	DTYPE	
52	000000006	00 AE		03	9F 00500 9F 00503 DD 00506 FB 00509 C7 00510	DIVL3	#3, DBG\$STA_TYP_ATOMIC #8, BITSIZE, LEN 72\$	1197
		03		50	DD 00506 FB 00509 C7 00510 11 00515 D1 00517 69	98: CMPL	72\$ RO, #3 70\$: 1197 : 1198
			48 30	AE	91 00510	BNEQ PUSHAB	DESCR	1198
	0000000G	00		02	DD 0051F FB 00522 DO 00529	PUSHL	TYPEID #2. DBG\$STA_TYP_DESCR	1
	44	50 AE 52	48 02	083 508 508 603 603	9A 0052D	MOVL MOVZBL MOVZWL	#2, DBG\$STA_TYP_DESCR DESCR, RO 2(RO), DTYPE	1198
		05			PB 00522 D0 00529 9A 0052D 3C 00532 11 00535 D1 00537 13 0053A D1 0053C 12 0053F D0 00541 71 3C 00545	BRB	(RO), LEN 72\$: 1198 : 1197
		07		50 05 50 09 0E 8F	D1 00537 70 13 0053A D1 0053C 12 0053F D0 00541 71 3C 00545	OS: CMPL BEQL CMPL	R0, #5 71\$	1198
	44			09	12 0053F	BNEQ IS: MOVL	RO, #7 72\$ #14, DTYPE	1198
		AE 52 50	2710		3C 00545 DO 0054A 72	MOVZWL	#10000, LEN	1198
		50 0E		AE 50	01 0054E 13 00551	CMPL	RO. #14	11177
		OF		21 50	D1 00553 13 00556	CMPL	RO. #15	1199
		10		50 17	01 00558 13 0055B	CMPL BEOL	RO, #16	1199
		11			D1 0055D 13 00560	CMPL BEQL	RO, #17	1199
		12		50 00	01 00562 13 00565	CMPL BEQL	RO #18	1199
		13		50	01 00567 13 0056A	CMPL BEQL	RO #19	1199
		14		50	D1 0056C 13 0056F 73	SS: BEQL	RO, #20	1199
	04	AB		0088	31 00571 8A 00574 74	S: BICB2	72\$ #14. DTYPE #10000, LEN DTYPE, RO RO, #14 74\$ RO, #15 74\$ RO, #16 74\$ RO, #17 74\$ RO, #18 74\$ RO, #19 74\$ RO, #20 74\$ 79\$	1200
5A	04	AB AB 54		05	DO 0054A 72 D1 0054E 13 00551 D1 00553 13 00556 D1 00558 D1 00562 13 00567 D1 00567	BISB2 MULL3	#2, 4(PRIMPTR) #12, SUBSCR_INDEX, R10	1200 1200 1200
		53 8F	04	50 120 00 50 00 00 00 00 00 00 00 00 00 00 00	D1 0056C 13 0056F 73 31 00571 8A 00574 74 88 00578 C5 0057C 9F 00580 D0 00584 D1 00587 15 0058E	PUSHAB	#1, 4(PRIMPTR) #2, 4(PRIMPTR) #12, SUBSCR_INDEX, R10 4(R10)[R7] a(SP)+, R3 R3, #524287	
	0007FFFF	8F		11	00 00584 01 00587 15 0058E	BEQL CMPL BEQL BRW BICB2 BISB2 MULL3 PUSHAB MOVL CMPL BLEQ	R3, #524287	

DBGPARSER V04-000				D 11 16-Sep-1984 02:1 14-Sep-1984 12:1	0:13 VAX-11 Bliss-32 V4.0-742 7:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 381 (46)
			53 DD 01 DD	00590 PUSHL 00592 PUSHL	R3 #1	; 1201 ; 1201
•	00000000G	000280F0 AB	01 DD 8F DD 03 FB 53 B0	00590 PUSHL PUSHL PUSHL PUSHL PUSHL PUSHL CALLS PUSHAB MOVL MOVAB CMPL BGTR TSTL BGTR TSTL BGTR TSTL BGTR PUSHL PUSHL PUSHL CALLS CMPL PUSHL PUSHL CALLS CMPL PUSHL PUSHL CALLS CMPL PUSHL PUSHL CALLS CMPL PUSHL	#164080 #3, LIB\$SIGNAL P3 16(PRIMPTR)	1
		5A 08	447 9F 9E DO	005A5 PUSHAB	#3, LIB\$SIGNAL R3, 16(PRIMPTR) 8(R10)[R7] a(SP)+, R10 -1(R10)[R3], R0	1201
		50 FF	AA43 9E	005AC MOVAB 005B1 CMPL	-1(R10)[R3], R0 R0, LEN	1201
			04 14 53 05	005B1 CMPL 005B4 BGTR 005B6 TSTL 005B8 BGTR 005BA 76\$: PUSHL	RO, LEN 76\$ R3 77\$	1201
		0408	52 00	005BA 76\$: PUSHL	//\$ LEN #^M <r3,r10></r3,r10>	1202
		00028008	52 DD 8F BB 03 DD 8F DD 05 FB	005BC PUSHR 005C0 PUSHL 005C2 PUSHL	#3 #164056	1202 1202 1202
	0000000G 00007FFF	00 8F	05 FB 5A D1	005C2 PUSHL 005C8 CALLS 005CF 77\$: CMPL	#5, LIB\$SIGNAL R10, #32767	1202
			11 15 5A DD	00506 BLEQ 00508 PUSHL	78\$ R10	1203
	0000000G	000280F8	01 DD 8F DD 03 FB	005DA PUSHL 005DC PUSHL 005E2 CALLS 005E9 78\$: MOVW	#1 #164088 #3, LIB\$SIGNAL R10, 18(PRIMPTR) 24(PRIMPTR), NODEPTR #1, 20(NODEPTR) #1, PICKED_UP_SUBSTRING SUBSCR_INDEX 80\$ #167792	1202
	12	AB 56 18	03 FB 5A BO AB DO	005E9 78\$: MOVW	R10, 18(PRIMPTR) 24(PRIMPTR), NODEPTR	1203
	14	A6 55	01 CE 01 DO	005F1 MNEGL 005F5 MOVL	#1, 20(NODEPTR) #1, PICKED_UP_SUBSTRING	1203
			54 D6 0D 11 8F DD	005F5 MOVL 005F8 INCL 005FA BRB 005FC 79\$: PUSHL	SUBSCR_INDEX 80\$	1203 1203 1203 1203 1203 1199 1205
	0000000G	00028F70 00 12 10	01 FB	00602 CALLS	#167792 #1, LIB\$SIGNAL TOOFEWSUB, 81\$ EXPECTED_SUBS	
		12 10	8F DD 01 FB AE E9 AE DD 01 DD	00609 80\$: BLBC 0060D PUSHL 00610 PUSHL	EXPECTED_SUBS	1205 1206
	0000000G	00028EA0 00 08	8F DD 03 FB	00612 PUSHL	#167584 #3, LIB\$SIGNAL	
54 01			00 ED	0061F 81\$: CMPZV	#0, #8, 1(R/), SUBSCR_INDEX 84\$	1206
	7E	54	55 C3 01 DD	0062B PUSHL	PICKED_UP_SUBSTRING, SUBSCR_INDEX, -(SP)	1206
		00028EB0	8F DD 0E 11 EF 9F	00655 BRB	#167600 83\$ P.AYS	1207
		00028362	01 DD 8F DD	0063B PUSHL	그는 🚜 📗 그리고 그렇게 보고 있는데 보고 있는데 회사를 하셨다. 전에 되는데 그리고 있는데 보다 되었다. 그리고 있다.	1207
	0000000G	00	03 FB	00643 835: CALLS	#164706 #3, LIB\$SIGNAL PRIMPTR, RO	1207
			5B 00	0064A 84\$: MOVL 0064D RET		1207

; Routine Size: 1614 bytes, Routine Base: DBG\$CODE + 3997

```
11993
11993
11994
11995
11996
11996
11996
11997
11998
11999
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
12000
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                12088
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                12089
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                12090
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  12091
12092
12093
12094
12095
12096
12097
12098
12099
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              12101
12102
12103
12104
12105
12106
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  12107
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  12108
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                12109
12110
12111
```

```
ROUTINE RESOLVE_COMPONENT (TYPEID, COMP_LIST, SYMID, PRIMPTR, COMPNAME) =
 FUNCTION
```

This routine is called from GET_RECORD_COMPONENT to resolve possible ambiguities where the user has specified X.Y and there is more than one record component named Y.

This situation arises in C, where membership checking is not enforced. Thus in C you can say X.Y even though Y is not a component in the record given by X. If we find more than one symid for Y, there is a possible ambiguity (although if all of the symids are record components with the same type and offset, then it is OK).

A different situation arises in BASIC. Here, we allow A::C to be an abbreviation for A::B::C (incomplete data qualification). In this case, the INCOMPLETE QUAL flag is lit. What we do here is call a routine which chases upscope pointers to determine whether we can get to the given TYPEID from the component SYMID.

INPUTS

- typeid for the record
- list of symids which are record components having
the same name. This list is in the form of TYPEID COMP_LIST a vector of longword, with the first longword

being the count.
address in which to leave the resolved symid,
if one is determined. SYMID

PRIMPTR - pointer to the input Primary - name of the component COMPNAME

OUTPUTS

The value TRUE is returned if a unique symid was determined; FALSE otherwise. If TRUE is returned then the output parameter SYMID is filled in.

BEGIN MAP

COMP_LIST: REF VECTOR[];

LOCAL FOUND:

DBG\$GL_CURRENT_PRIMARY = .PRIMPTR;

If incomplete data qualification is allowed in this language (for example A:: C in place of A:: B:: C) then we search the list of candidate components to see if there is a unique one in the given record. For example, if the user says A:: C, and there are several components (, but only one belongs in record A, then that is the one we want.

This is the code path taken for language BASIC.

IF .INCOMPLETE_QUAL THEN

BEGIN

FOUND = FALSE;

RETURN FALSE:

END;

						1	-Sep-19 4-Sep-19	84 02:10 84 12:17	2:13 VAX-11 Bliss-32 V4.0-742 2:30 [DEBUG.SRC]DBGPARSER.B32:1	Page 384 (47)
	0000000G	00	00000000					-COMPONE WORD MOVL BLBC CLRQ BRB	Save R2.R3 PRIMPTR, DBG\$GL_CURRENT_PRIMARY INCOMPLETE_QUAL, 3\$ I	: 1207 : 1212 : 1213 : 1214
DB	E2FB OC	CF 007 BC 5522		50 53 BC42	D9C14DDDB980038DDDB	00002 00001 000013 000015 000015 000010 000029 000035 000035 000042 000058 00005 00005 00005 00005	2\$:	WORD MOVL BLBC CLRQ BRB CLRQ BRB CLRQ BLBSHL PUSHL CALLS BLBS MOVL AOBLEQ BUSHL PUSHL	PRIMPTR TYPEID aCOMP LIST[I] #4. CRECK_UPSCOPE R0. 2\$ FOUND, 5\$ aCOMP_LIST[I], aSYMID #1, FOUND aCOMP_LIST, I, 1\$ FOUND, 4\$ COMPNAME	1214 1215 1215 1214 1216 1216
	00000000G	00 50 01 BC 50	14 00028¢80 08 04	01 B53 01 87 08 08 08 09 00	DD	0003D 00040 00042 00048 0004F 00055 00058 0005A	3\$:	PUSHL PUSHL CALLS BRB MOVL CMPL BNEQ MOVL	#167040 #3, LIB\$SIGNAL 4\$ COMP_LIST, RO (RO), #1 5\$ 4(RO), @SYMID	1216 1217 1217 1218 1218
		50		50	04 04 04	00062	4\$: 5\$:	MOVL RET CLRL RET	#1, RÔ RO	1218

; Routine Size: 102 bytes, Routine Base: DBG\$CODE + 3FE5

ROUTINE SAVE_SUBSCRIPTS(PATHDESC, SUBSCR_DESC): NOVALUE =

When parsing PLI or BASIC Primaries we want to save away the subscripts that we see along the way. This routine picks up subscripts and saves them into the SUBSCR_DESC data structure.

INPUTS

PATHDESC - A pointer to the pathname descriptor for the pathname we have parsed so far.

SUBSCR_DESC - A data structure containing the subscript

values.

OUTPUTS

BEGIN

PATHDESC: REF PTH\$PATHNAME, SUBSCR_DESC: REF SUBSCR\$DESC;

DECLTYPE: REF DBG\$VALDESC,

LA PTR: REF VECTOR[,BYTE], LOW_RANGE_VAL, PATH_INDEX, SAVED_RADIX, SUBSCR_COUNT, THIS_SUBSCR_IS_RANGE,

VALADOR: REF VECTOR[,LONG], VALPTR: REF DBG\$VALDESC; Pointer to Value Descriptor for declared subscript data type Lookahead pointer into input Low value of a subscript range Count of pathname components Temporarily saved expression radix Actual subscript count in input line flag set if the current subscript is given as a subscript range Lexical Token Pointer to integer subscript value Pointer to subscript Value Descriptor

Note how many subscripts we have previously picked up (i.e., in earlier calls to this routine, while parsing this same expression). In PL/I, for example, the subscripts may arrive in separate pieces, and do not necessarily have to be associated with the "right" component, e.g., X(1,2).Y(3).Z(4,5)

This routine also picks up substring references in COBOL. Because of this, we set a marker to indicate where we were in SUBSCR DESC when we entered this routine. E.g., in the COBOL expression X(1,2,3)(1:5) we set a marker to indicate that the 1:5 came in a separate set of parenthesis. It would be illegal otherwise, so this marker can be used later to decide whether to signal an error.

SUBSCR_COUNT = .SUBSCR_DESC[0, SUBSCR\$B_SUBCNT]; SUBSCR_DESC[.SUBSCR_COUNT, SUBSCR\$V_MARKER] = TRUE;

Loop through the subscript expressions for this array reference. Each subscript is parsed, evaluated, and converted to integer. Its value is then stored in the SUBSCR_DESC data structure.

PATH_INDEX = .PATHDESC[PTH\$B_TOTCNT];

Page 386 (48)

```
12255
12256
12257
12258
12259
12261
12262
12263
12263
12265
12266
12267
12270
12271
12272
```

```
TERMINATOR_CODE and TERMINATOR_LENGTH as a side-effect.
SAVED_RADIX = .EXPRESSION_RADIX;

EXPRESSION_RADIX = DBG$K_DECIMAL;

VALPTR = DBG$EXPRESSION_PARSER (FALSE, .SUBSCRIPT_TERM_TBL);

EXPRESSION_RADIX = .SAVED_RADIX;
```

Check the terminator code. If there was no terminator (the input line just ended), signal an error. Otherwise we got a comma or closing subscript parenthesis and we increment CHARPIR to get past it.

IF .TERMINATOR CODE EQL TOKENSK TERM NONE THEN SIGNAL (DBG\$_MISCLOSUB); CHARPTR = .CHARPTR + .TERMINATOR_LENGTH;

We now need to convert the subscript to one of the appropriate dtype. We need to set up a target descriptor for the conversion routine. We allocate a skeleton descriptor and fill in some of the fields.

```
DECLTYPE = DBG$MAKE_SKELETON_DESC(DBG$K_VALUE_DESC, 4);
DECLTYPE[DBG$B_DHDR_KIND] = RST$K_DATA;
DECLTYPE[DBG$B_DHDR_FCODE] = RST$K_TYPE_ATOMIC;
DECLTYPE[DBG$B_VALUE_CLASS] = DSC$K_CLASS_S;
DECLTYPE[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
DECLTYPE[DBG$W_VALUE_LENGTH] = 4;
DECLTYPE[DBG$W_VALUE_POINTER] = DECLTYPE[DBG$A_VALUE_ADDRESS];
```

finally call the conversion routine. This routine checks that the conversion is legal before doing it.

VALPTR = DBG\$EVAL_LANG_OPERATOR(DBG\$GL_CONVERT_TOKEN, .VALPTR, .DECLTYPE);
VALADDR = .VALPTREDBG\$C_VALUE_POINTER];

If the terminator at the end of this subscript expression was a colon we have a subscript range (for example, ''ARR(1:5,2)''). We thus set the subscript-range flag and save the low value of the range, i.e. the value we just picked up. If this is the first range in the subscript list, we also turn all previous subscripts into ranges by setting the lower and upper bound for each such subscript to the corresponding subscript value. This in effect defines a new array which constitutes a "slice" of the original array.

.TERMINATOR_CODE EQL TOKENSK_TERM_COLON

BEGIN IF .THIS_SUBSCR_IS_RANGE THEN SIGNAL(DBG\$_INVRANSPEC);
THIS_SUBSCR_IS_RANGE = TRUE;
LOW_RANGE_VAL = .VALADDR[0];

! The terminator was not a colon, so we now have the full subscript ! specification. Fill the subscript value into the Array Sub-Node's

```
DBGPARSER
V04-000
                                                                                                                                            VAX-11 Bliss-32 V4.0-742 [DEBUG.SRC]DBGPARSER.B32:1
                                                                         subscript vector. Set up the bounds for an array "slice" if this or any previous subscript specification in this array reference
                                                                         consisted of a subscript range. Also bump the subscript count.
 12277
12278
12279
12280
12281
12283
12288
12288
12288
12288
12288
12289
12298
12298
12298
12298
12298
12306
12308
12318
12318
12318
12318
                                                                      ELSE
                                                                            BEGIN
                                                                                If this subscript is specified as a subscript range, check that
                                                                               the first value in the range is not greater than the second. Also clear the subscript-is-range flag for the next subscript.
                                                                             IF .THIS_SUBSCR_IS_RANGE
                                                                             THEN
                                                                                   BEGIN
                                                                                   SUBSCR_DESCE.SUBSCR_COUNT, SUBSCR$V_RANGE] = TRUE;
THIS_SUBSCR_IS_RANGE = FALSE;
                                                                               Otherwise, set the low range value to be the subscript value.
                                                                            ELSE
                                                                                   LOW_RANGE_VAL = .VALADDR[0];
                                                                               finally fill in the subscript value itself (the start of the range), increment the subscript count, and loop.
                                                                            SUBSCR_DESC[.SUBSCR_COUNT, SUBSCR$B_PATH_INDEX] = .PATH_INDEX;
SUBSCR_DESC[.SUBSCR_COUNT, SUBSCR$L_LBOUND] = .LOW_RANGE_VAL;
IF .SUBSCR_DESC[.SUBSCR_COUNT, SUBSCR$V_RANGE]
                                                                             SUBSCR_DESC[.SUBSCR_COUNT, SUBSCR$L_UBOUND] = .VALADDR[0];
SUBSCR_COUNT = .SUBSCR_COUNT + 1;
                                                                            END:
                                                                      END:
                                                                END:
                                                                                                                   ! End of WHILE loop over subscripts
                                                            We have picked up all the subscripts within this set of subscript paren-
                                                          SUBSCR_DESC[O, SUBSCR$B_SUBCNT] = .SUBSCR_COUNT;
                                                         RETURN:
                                                         END:
                                                                                        OFFC 00000 SAVE_SUBSCRIPTS:
                                                                                                                      .WORD
                                                                                                                                                                                                           1218
                                                                                                                                   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
                                                                                                                                  #8, SP
SUBSCR_DESC, R3
1(R3), SUBSCR_COUNT
#12, SUBSCR_COUNT, RO
(RO)[R3]
                                                                                           C2
D0
9A
C5
9F
                                                                                                                                                                                                            1223
                                                                                                                      MOVL
                                                                                                00009
                                                                                                                      MOVZBL
                                                                                                                                                                                                            1223
                                        50
                                                                                                                      MULL3
                                                                                                                      PUSHAB
```

					12	-Sep-198	84 02:10 84 12:17	VAX-11 Bliss-32 V4.0-742 DEBUG.SRCJDBGPARSER.B32:1	Page 389 (48)
	00000000	9E 59 EF 02	00040000 8F 04 BC 58 01 00000000 EF 03	D4 D0 D1	00014 0001B 0001F 00021 00028	1\$:	BISL2 MOVZBL CLRL MOVL CMPL BNEQ	#262144, a(SP)+ aPATHDESC, PATH_INDEX THIS_SUBSCR_IS_RANGE #1, TERMINATOR_CODE TERMINATOR_CODE, #2	1224 1224 1224 1224
		56 20	00000000 0181 66 04 56 7	D1 12 31 09 12 06 11	0002F 00031 00034 0003B 0003E 00040	2\$: 3\$:	BRW MOVL CMPB BNEQ INCL BRB CMPB	2\$ 17\$ CHARPTR, LA_PTR (LA_PTR), #32 4\$ LA_PTR 3\$	1225
		2A	66 03 0094	91	00044	48:	CMPB BEQL	(LA_PTR), #42	1225
	00000000	EF	01 A6 7E 00000000 EF	13 31 9E 04 00 7C	00040 00042 00044 00047 00049 00056 00056 0005E 00063	5\$:	BRW	1(R6), CHARPTR -(SP) SUBSCRIPT_TERM_TBL -(SP)	1226 1227 1227 1227
	C9F0	CF 6E 50 50	01 A6 7E 00000000' EF 7E 04 50 00000000' EF	9E	0005E 00063 00066 0006D		CLRL PUSHL CLRQ CALLS MOVL MOVAB CMPL BEQL	#4, DBG\$LEXICAL_SCANNER RO, TOKEN TERMINATOR_TOKEN, RO TOKEN, RO	1227
	04 05	AE AE	01	90 90 9F DD	00070 00072 00076 00081 00083 00089 00090 00097		MOVB MOVB PUSHAB	6\$ #1, ASCIC_STRING aCHARPTR, ASCIC_STRING+1 ASCIC_STRING #1	1227 1227 1227
	0000000G	00	000289E2 8F 00000000' EF 0D	DD FB D1 12	00083 00089 00090 00097	6\$:	PUSHL PUSHL CALLS CMPL BNEQ PUSHL	#166370 #3, LIB\$SIGNAL TERMINATOR_CODE, #3 7\$	1228
	0000000G	00	00028F08 8F 00000000' EF 0D	U 3	UUUMD	7\$:	PUSHL CALLS TSTL BNEQ	#167688 #1, LIB\$SIGNAL TERMINATOR_CODE 8\$	1228 1228
50	000000000	00 EF 54	00000000' FF 04 AE 01 000289E2 8F 03 00000000' EF 00 00028F08 8F 01 00000000' EF 00 00028E90 8F 01 00000000' EF	DD	000AC 000AE 000B4 000BB 000C6	8\$:	PUSHL	#167568 #1 IRSSIGNAL	1228 1228 1229
	02 A	043 9E 043	00020000 6043 01 01 0000	90 9F C8 88	000CE 000D1 000D8		PUSHAB BISL2 BISB2 BRW	TERMINATOR LENGTH, CHARPTR #12, SUBSCR_COUNT, RO PATH_INDEX, (RO)[R3] (RO)[R3] #131072, a(SP)+ #1, 2(RO)[R3] 15\$	1229
	00000000	5B EF	00000000° EF	C88 31 D0 D0 D4	000CE 000D1 000D8 000DD 000E0 000E7 000EE	9\$:	MOVL MOVL PUSHL CLRL	EXPRESSION_RADIX, SAVED_RADIX #10, EXPRESSION_RADIX SUBSCRIPT_TERM_TBL -(SP)	1229 1229 1230 1230 1230
	00000000°	CF 55 EF	00000000' EF 7E 02 50 5B 00000000' EF 0D 00028E90 8F 01 00000000' EF	DO DO D5 12	000FB 000FE 00105 0010B		CALLS MOVL MOVL TSTL BNEQ PUSHL	#2. DBG\$EXPRESSION_PARSER RO, VALPTR SAVED_RADIX, EXPRESSION_RADIX TERMINATOR_CODE 10\$	1230 1231
	000000000	00 EF	00028E90 8F 00000000 EF	FB CO	0010 <u>0</u> 00113 0011A	10\$:	PUSHL CALLS ADDL2	#167568 #1, LIB\$SIGNAL TERMINATOR_LENGTH, CHARPTR	1231

DBGPARSER V04-000			M 11 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 390 (48)
	00000000G 7E 00 06 A2 14 A2 18 A2	7A 8F 02 50 0602 8F 01080004 8F 20 A2 52	DD 00125 9A 00127 FB 0012B CALLS	; 1232 ; 1232 ; 1232 ; 1233 ; 1233
	00000000G 00 55 57 03	00000000° EF 03 50 18 A5	DD 0014A PUSHL VALPTR 9F 0014C PUSHAB DBG\$GL CONVERT TOKEN FB 00152 CALLS #3, DBG\$EVAL_LANG_OPERATOR DO 00159 MOVL RO, VALPTR DO 0015C MOVL 24(VALPTR), VALADDR D1 00160 CMPL TERMINATOR_CODE, #3 12 00167 BNEQ 12\$	1233
	00000000G 00 58 5A	00028F08 8F 01 01	D1 00160	1235
	50 02 A043	67 31 58 00 01 58	11 00186 PDP 1/6	1235 1235 1234 1237 1237
	50 5A 54 6043	04 A043	00 00191 13%: MOVL (VALADDR), LOW RANGE VAL C5 00194 14\$: MULL3 #12, SUBSCR_COUNT, RO 90 00198 MOVB PATH_INDEX, (RO)[R3] 9F 0019C PUSHAB 4(RO)[R3]	1237 1257 1238 1238
	07 02 A043 9E	08 A043	DO 001AD MOVL (VALADDR), a(SP)+	1239 1239
	01 A3	FE73	D6 001B0 15\$: INCL SUBSCR_COUNT 31 001B2 16\$: BRW 1\$ 90 001B5 17\$: MOVB SUBSCR_COUNT, 1(R3) 04 001B9 RET	1239 1224 1240 1240

; Routine Size: 442 bytes, Routine Base: DBG\$CODE + 404B

ROUTINE SCAN_QUOTED_STRING(TOKENBUFFER, TOKEN_TYPE): NOVALUE = **FUNCTION**

This routine scans a quoted character string and returns the found string in Counted ASCII format to a caller-provided buffer. It expects the OWN pointer CHARPTR to point to the quote character at the start of the character constant and it assumes that this in fact is a valid quote character. It then scans for the closing quote character (which must be the same character as the opening quote), treating doubled up quote characters within the string as a single quoted quote character. If the language is set to PL/I, then after finding the closing quote, it searches for a 'B' or 'b', to determine if the string could be a bit-string rather than a character string. If the quote characters are not ''', or if any double quotes are encountered, then the trailing 'B' is not searched for. searched for.

If a carriage-return (end of input line) is found before the closing quote or if the string exceeds 255 characters (the longest Counted ASCII allows), an error is signalled. Otherwise, CHARPTR is left pointing to the first character after the closing quote and the string itself is returned as Counted ASCII to the caller's buffer.

INPUTS

TOKENBUFFER - A pointer to the buffer in which the character string is to be accumulated. This buffer is expected to be 256 characters long, enough for the longest Counted ASCII string.

TOKEN_TYPE - Address of where to return token's type.

OUTPUTS TOKENBUFFER - The quoted character string is accumulated and returned as a Counted ASCII string to the buffer pointed to by TOKENBUFFER.

TOKEN_TYPE - Is filled in with either TOKEN\$K_STRING or TOKEN\$K_BIT_STRING.

BEGIN

TOKENBUFFER: REF VECTOR[,BYTE], ! Pointer to buffer for char string TOKEN_TYPE : REF VECTOR[1]; ! Longword to receive token's type.

LOCAL QUOTE.

DOUBLED_QUOTES,

TOKENLEN:

Quote character which started the current quoted string constant flag denotes whether or not there exist doubled up quotes (''). Length of quoted string so far

We pick up the closing quote character (which must be the same as the opening quote character) and then scan for the end of the string. Doubled up quotes are reduced to a single quote within the string

```
VAX-11 Bliss-32 V4.0-742
EDEBUG.SRCJDBGPARSER.832:1
    and a carriage-return (end of line) is treated as an error (quotes not balanced). Doubled up quotes set DOUBLED_QUOTES flag.
DOUBLED_QUOTES = FALSE;
QUOTE = .CHARPTR[0];
TOKENLEN = 0;
TOKEN_TYPE[0] = TOKEN$K_STRING;
WHILE TRUE DO BEGIN
        CHARPTR = .CHARPTR + 1;
IF .CHARPTR[0] EQL CAR RET THEN SIGNAL (DBG$_MATQUOMIS);
IF .CHARPTR[0] EQL .QUOTE
        THEN
                BEGIN
                CHARPTR = .CHARPTR + 1;
IF .CHARPTR[0] NEQ .QUOTE THEN EXITLOOP;
                DOUBLED_QUOTES = TRUE;
        IF .TOKENLEN GEQ 255 THEN SIGNAL(DBG$_QUOSTRLONG);
TOKENLEN = .TOKENLEN + 1;
TOKENBUFFER[.TOKENLEN] = .CHARPTR[0];
        END:
   If language is PL/I, then check for a bit-string. If the quotes are "" and there were no doubled up quotes, then see if the next character is the letter "B" or "b". If so, then change TOKEN_TYPE to be a bit-string (TOKEN$K_BIT_STRING). The "B" is not part of the length of the string.
IF .DBG$GB_LANGUAGE_EQL_DBG$K_PLI
AND .QUOTE EQL_DBG$K_QUOTE
AND NOT .DOUBLED_QUOTES
        IF .CHARPTR[0] EQL %C'B' OR .CHARPTR[0] EQL %C'b'
        THEN
                BEGIN
                CHARPTR = .CHARPTR + 1;
TOKEN_TYPE[0] = TOKEN$K_BIT_STRING;
                END:
   We found the end of the string. Complete the Counted ASCII string in the TOKENBUFFER buffer by filling in the length and return.
TOKENBUFFER[0] = .TOKENLEN;
RETURN;
```

END:

DBGPARSER V04-000	C 12 16-Sep-1984 02:10:13 VAX-11 Bliss-32 V4.0-742 14-Sep-1984 12:17:30 [DEBUG.SRC]DBGPARSER.B32;1	Page 393 (49)
53 53 53 : Routine Size:	Section Sect	1246 1246 1246 1246 1247 1247 1247 1247 1247 1248 1248 1248 1248 1249 1249 1249 1249 1249 1250 1250 1250 1250
	.EXTRN LIB\$SIGNAL PSECT SUMMARY	
Name	Bytes Attributes	
DBG\$GLOBAL DBG\$OWN DBG\$PLIT	4 NOVEC, WRT, RD , NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2) 1344 NOVEC, WRT, RD , NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2) 13389 NOVEC, NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC, ALIGN(0)	

VAX-11 Bliss-32 V4.0-742 [DEBUG.SRCJDBGPARSER.B32;1

Page 394 (49)

D

; DBG\$CODE

17037 NOVEC, NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC, ALIGN(0)

Library Statistics

file	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1 _\$255\$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1 _\$255\$DUA28:[DEBUG.OBJ]DBGLIB.L32;1 _\$255\$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1	18619 32 1545	36 490	0 6 31	1000 7 97	00:02.0 00:00.1 00:02.0
_\$255\$DUA28: [DEBUG.OBJ]DBGMSG.L32;1 _\$255\$DUA28: [DEBUG.OBJ]DBGGEN.L32;1	418 386 150	125 57 1	29 14 0	31 22 12	00:00.4 00:00.3 00:00.3

: Information: : Warnings: : Errors:

;

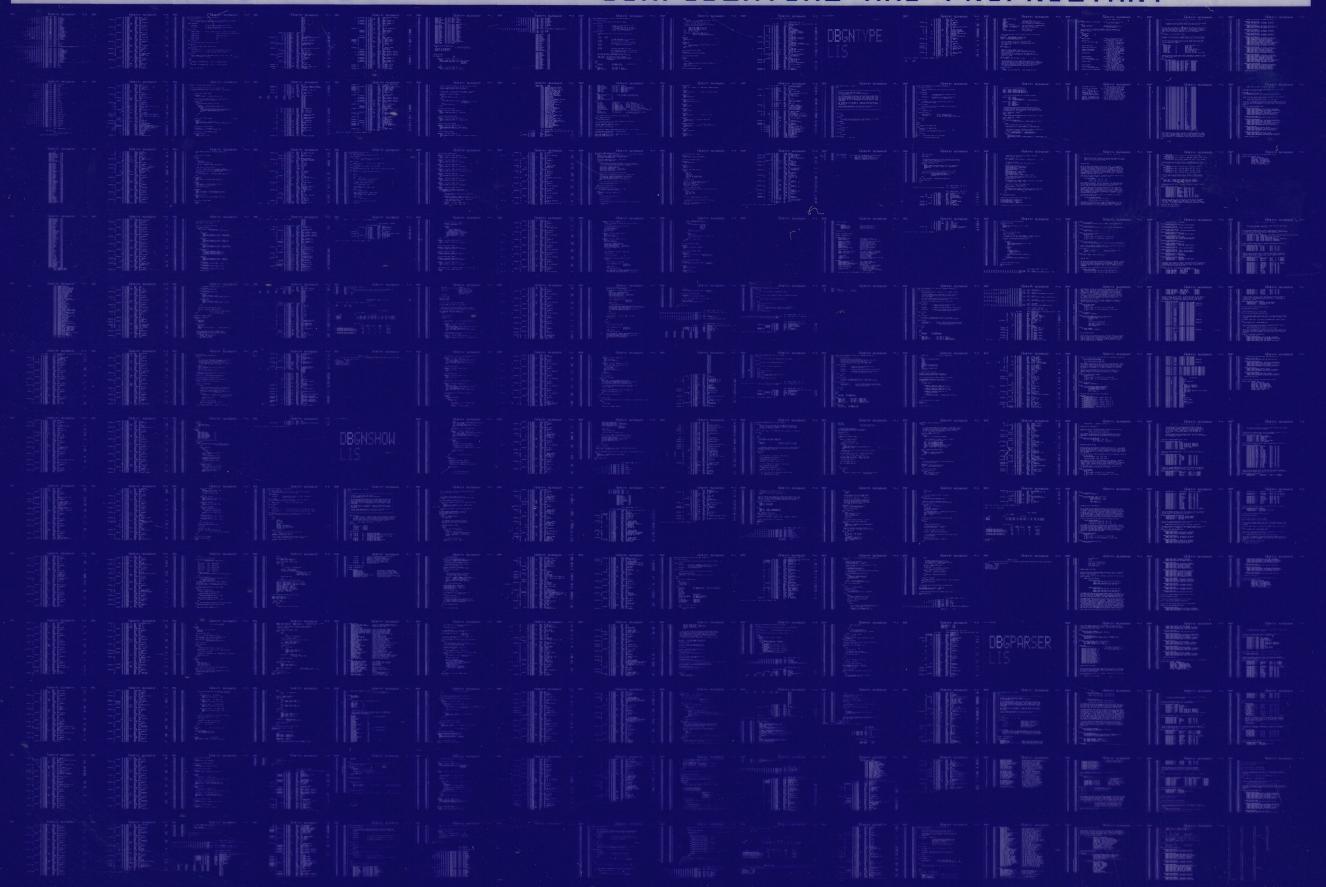
COMMAND QUALIFIERS

BLISS/CHECK=(FIELD, INITIAL, OPTIMIZE)/LIS=LIS\$:DBGPARSER/OBJ=OBJ\$:DBGPARSER MSRC\$:DBGPARSER/UPDATE=(ENH\$:DBGPARSER)

; Size: 17037 code + 14737 data bytes ; Run Time: 07:23.7 ; Elapsed Time: 17:24.9 ; Lines/CPU Min: 1692 ; Lexemes/CPU-Min: 28123 ; Memory Used: 1461 pages ; Compilation Complete

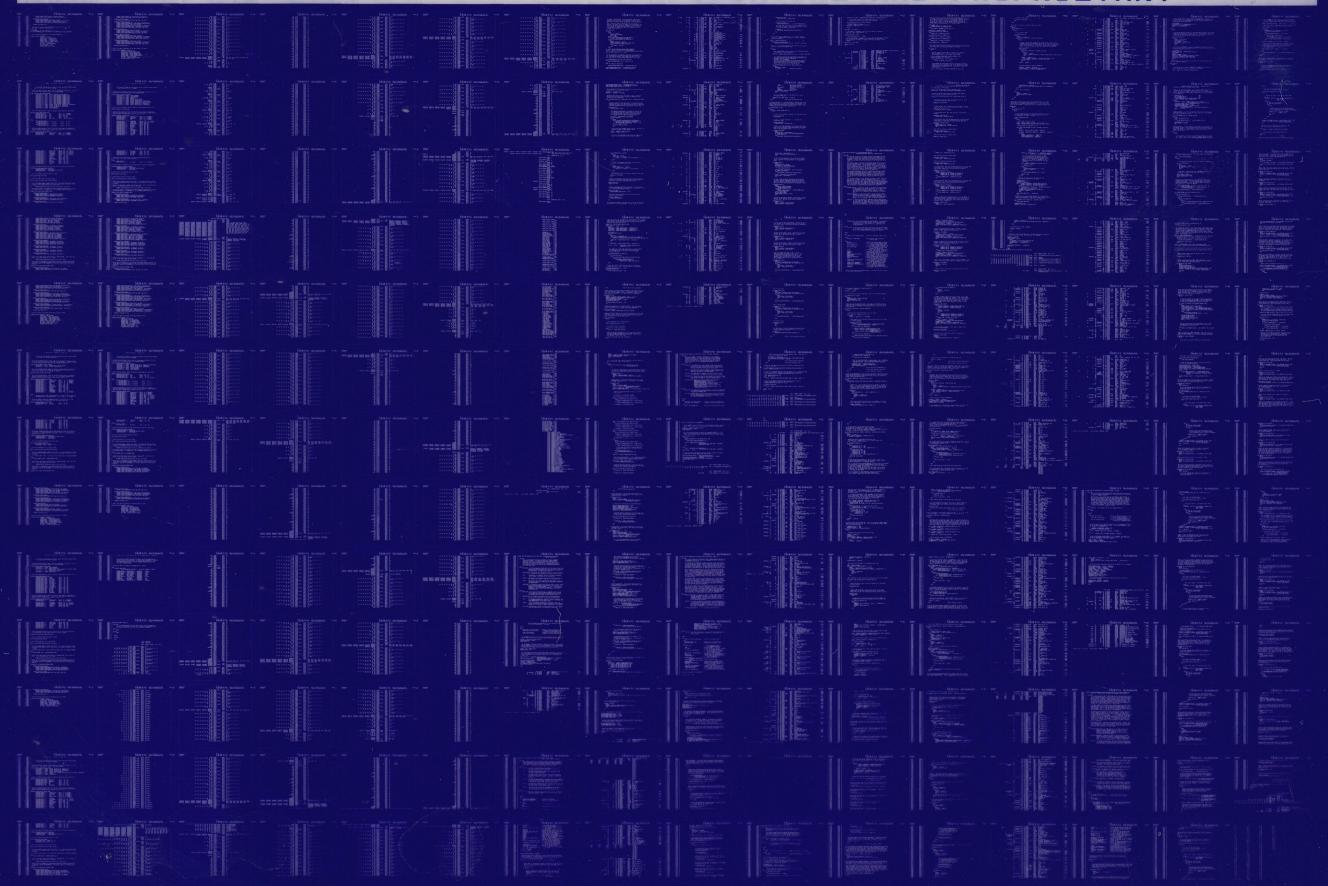
0089 AH-BT13A-SE

DIGITAL EQUIPMENT CORPORATION CONFIDENTIAL AND PROPRIETARY



0090 AH-BT13A-SE

DIGITAL EQUIPMENT CORPORATION CONFIDENTIAL AND PROPRIETARY



0091 AH-BT13A-SE

DIGITAL EQUIPMENT CORPORATION CONFIDENTIAL AND PROPRIETARY

